AFRL-RI-RS-TR-2009-103
Final Technical Report
April 2009

CASE MASTER

Techteam Government Solutions, Inc.

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

STINFO COPY

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

NOTICE AND SIGNATURE PAGE

Using Government drawings, specifications, or other data included in this document for any purpose other than Government procurement does not in any way obligate the U.S. Government. The fact that the Government formulated or supplied the drawings, specifications, or other data does not license the holder or any other person or corporation; or convey any rights or permission to manufacture, use, or sell any patented invention that may relate to them.

This report was cleared for public release by the 88th ABW, Wright-Patterson AFB Public Affairs Office and is available to the general public, including foreign nationals. Copies may be obtained from the Defense Technical Information Center (DTIC) (http://www.dtic.mil).

AFRL-RI-RS-TR-2009-103 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE DIRECTOR:

/s/ /s/

NANCY A. ROBERTS
Work Unit Manager

JOSEPH CAMERA, Chief
Information & Intelligence Exploitation Division
Information Directorate

This report is published in the interest of scientific and technical information exchange, and its publication does not constitute the Government's approval or disapproval of its ideas or findings.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
APR 09	Final	Aug 06 – Jan 09

4. TITLE AND SUBTITLE

CASE MASTER

5a. CONTRACT NUMBER
FA8750-06-C-0193

5b. GRANT NUMBER
N/A

5c. PROGRAM ELEMENT NUMBER
N/A

6. AUTHOR(S)

Rafael Alonso, Philip Bramsen, Sven Brueckner, Liz Downs, Hua Li, Van Parunak, and Andrew Yinger

5d. PROJECT NUMBER
CASE

5e. TASK NUMBER
00

5f. WORK UNIT NUMBER
02

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Techteam Government Solutions, Inc.
3863 Centerview Drive, Suite 150
Chantilly, VA 20151-3287

8. PERFORMING ORGANIZATION REPORT NUMBER
N/A

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

AFRL/RIED
525 Brooks Rd.
Rome NY 13441-4505

10. SPONSOR/MONITOR'S ACRONYM(S)
N/A

11. SPONSORING/MONITORING AGENCY REPORT NUMBER
AFRL-RI-RS-TR-2009-103

12. DISTRIBUTION AVAILABILITY STATEMENT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED. PA# 88ABW-2009-1388

13. SUPPLEMENTARY NOTES

14. ABSTRACT

The CASE MASTER project focused primarily on the development of the InformANTS prototype, a distributed, scalable, and adaptive system that tracks the activities of one or more intelligence analysts, maintains up-to-date models of these analysts' interest, models dynamically changing sets of documents from the web or in InformANTS-enabled wikis, and recommends at any time relevant documents or other analysts to a given user. In the development of the prototype, the team engaged in necessary research in the domains of user modeling, self-organizing data, management of collaboration processes, value of information to users, and information contents in models of documents. The remainder of this report presents our progress towards this goal, which culminated in a successful evaluation of a fully integrated and complex prototype in a National Institute of Standards and Technology exercise in September 2008.

15. SUBJECT TERMS

Self organizing data, collaboration, document modeling, user modeling, adaptive, information pull, information push, information matching, user interest modeling, information wiki

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Nancy A. Roberts
U	U	U	UU	101	19b. TELEPHONE NUMBER (*Include area code*) N/A

TABLE OF CONTENTS

LIST OF FIGURES

1 PROJECT SUMMARY

The CASE MASTER project's main objective was to develop a prototype of the InformANTS system. The remainder of this report presents our progress towards this goal, which culminated in a successful evaluation of a fully integrated and complex prototype in a NIST exercise in September 2008.

What is InformANTS?—InformANTS (Information Agents for Networked Teaming Support) turns passive information into active information packets (InfoPacks) that know their own contents, organize with other InfoPacks that might complement their contents, and seek out users who need them.

Before computers, information depended on people to file, retrieve, transport, interpret, and act on it. Even today, applications such as the web (Figure 1a) and email (Figure 1b), still treat information as passive. InformANTS' novel blend of artificial intelligence and artificial life makes InfoPacks active (Figure 1c), providing an innovative information repository and knowledge exploration tool.

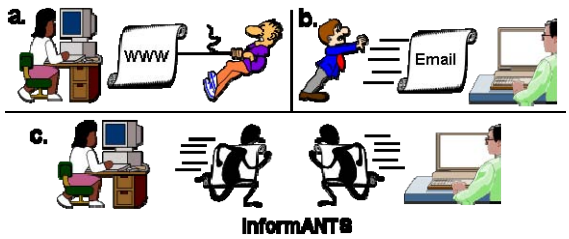

Figure 1: Passive (a, b) vs. active (c) information

Where did it come from?—NIMD's Ant CAFÉ demonstrated that analyst queries were sufficient to maintain a cognitive model of the analyst's interests and hypotheses. This model modulated a swarming mechanism that retrieved documents of interest. Combining symbolic and sub-symbolic reasoning mechanisms in a single system was a radical innovation. InformANTS extends Ant CAFÉ to make information active.

Representations.—InformANTS uses a *satisficing* approach to knowledge representation. Optimal processing of a full ontology requires human intelligence, while the minimal concept maps used by the Ant CAFÉ leave out much useful information. InformANTS works towards an intermediate approach, combining limited relations *within* geospatial maps, social networks, and hierarchical task networks with *between*-network relations based on case grammar (as a first approximation, InformANTS establishes relations based on co-occurrence of concepts in text). Incoming InfoPacks *match* themselves in the system, *discover* other nearby InfoPacks, and *deliver themselves* to users corresponding to nearby user InfoPacks. As user InfoPacks discover one another, they form virtual interest groups.

Utilization.—InformANTS can enhance many interfaces, including social networking and email. We demonstrated it as an "active wiki," enabling rapid transition as an extension to Intellipedia. The population of InfoPacks includes both wiki pages and user models. User browsing and editing enable the formation of user models that guide the movement of InfoPacks through InformANTS, enabling the wiki to suggest new linkages among pages, page recommendations for users, and virtual interest groups that improve the efficiency and speed of other processes.

2 SCIENTIFIC AND TECHNICAL RESULTS AND ACCOMPLISHMENTS

The CASE MASTER project focused primarily on the development of the InformANTS prototype, a distributed, scalable, and adaptive system that tracks the activities of one or more intelligence analysts, maintains up-to-date models of these analysts' interest, models dynamically changing sets of documents from the web or in InformANTS-enabled wikis, and recommends at any time relevant documents or other analysts to a given user. In the development of the prototype, the team engaged in necessary research in the domains of user modeling, self-organizing data, management of collaboration processes, value of information to users, and information contents in models of documents.

The following major sections detail the architecture and operation of the InformANTS prototype, insights gained in the associated research, and results from significant evaluation exercises.

2.1 THE INFORMANTS PROTOTYPE

Over the course of the project, we developed a series of successively refined InformANTS prototypes that provided growing capabilities in various CASE integration demonstrations. The final prototype that was deployed in the September 2008 NIST evaluation had the highest sophistication in terms of capabilities and scalability. Therefore, we present the following architecture and component functionalities of this prototype.

2.1.1 ARCHITECTURE

InformANTS is designed to provide complex products to other CASE program components via webservices. Internally, InformANTS runs stand-alone components connected in an xmlBlaster communications architecture for more simple but also more orders-of-magnitude more frequent interactions. The internal component integration supports the scalability and responsiveness requirements of the experimental design of the NIST evaluation. Conceptually, the InformANTS prototype follows what we call the "Hyper-Spectral Modeling" approach.

In the following, we first describe the modeling approach and the key internal data structure – the InfoPack. Then we introduce the main InformANTS components define the webservices provided in the NIST evaluation prototype and finally, we specify the main internal interaction protocols.

2.1.1.1 HYPER-SPECTRAL MODELING

The InformANTS system ingests a stream of content-carrying entities (e.g., analytic events associated with user actions, pointers to documents recommended by Google in response to queries). Each of these entities are modeled using one or more modeling techniques that turn raw content into "object models", so for each discrete piece of content InformANTS creates a set of object models. Furthermore, for any content that is associated with a user action (e.g., page viewed, text copied, query issued), the User Modeling System (UMS) in InformANTS will first create and then subsequently update a model of the user. InformANTS then organizes the models of external documents (e.g., a page viewed by a user, a document recommended by Google) and models of the users that triggered analytic events in a common information space to match similar documents and users.

The purpose of the InformANTS system is to match users and documents to provide various recommendations (e.g., recommend relevant documents to a user based on the current interest, recommend other users for collaboration, or reorder documents recommended by Google to reflect the larger context of the user interest). To maximize the resolution of the information that is available to this matching process, InformANTS avoids premature merging of models for the same document content or the same user that are based on different modeling approaches that may have different emphases. We draw the analogy to hyper-spectral imaging (Figure 2), where the same scene is recorded separately for different frequency bands that are stored separately for the image-analysis process. Prematurely merging these images before the analysis would lose relevant information.

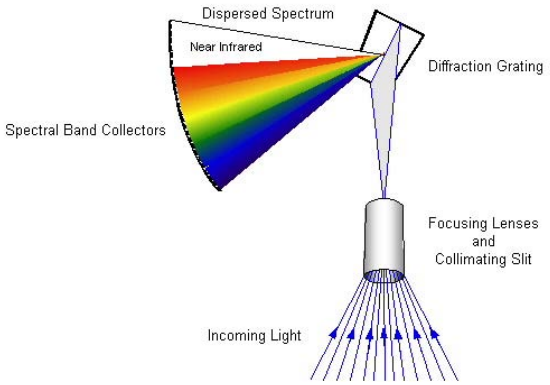

Figure 2. Schema of Hyper-Spectral Imaging.

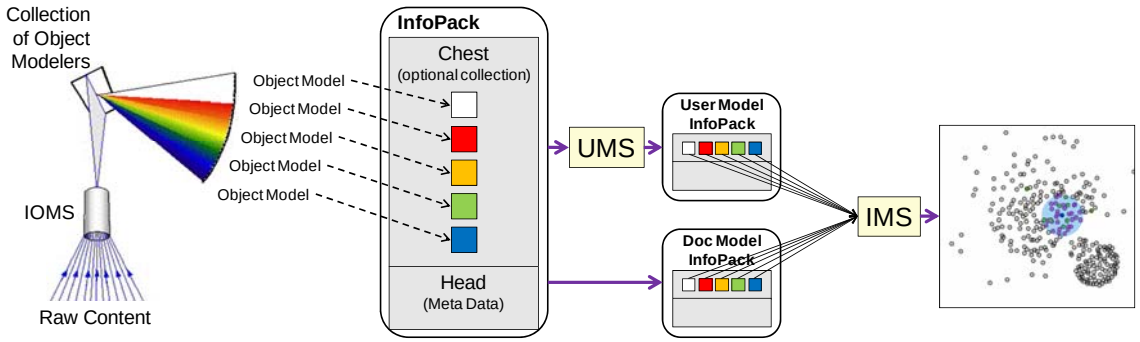

Figure 3. InfoPacks carry different object models for the same content, where "content" is either an external document (handled directly) or the aggregation of external content associated with a particular user (formed by the UMS). Only the IMS merges these hyper-spectral models in the matching process.

InformANTS follows the hyper-spectral modeling approach. Any content that is presented to the system is turned into multiple object models by the collection of Object Modeler instances in the Information Object Modeling System (IOMS). If the content is associated with an external document, the resulting object models are stored in a data structure that we call an InfoPack, which is then InformANTS' representation of that document (Figure 3). Similarly, object models associated with an analytic event are used by the UMS to define models of the current user interest, which, again, are stored as InfoPacks. Following the analogy of the hyper-spectral imaging, the image processing (here the Information Matching System (IMS)) analyzes all images from all frequency bands (here object models) to create its product (here relevance-matching of documents and users).

As a consequence of the hyper-spectral modeling approach, InformANTS maintains instances of object modelers for different modeling approaches in the IOMS (section 2.1.2.3), user modeling techniques based on the different approaches (section 2.1.3), and object model reasoners that influence the matching process for a specific approach (section 2.1.4.1).

2.1.1.2 INFORMANTS-INTERNAL INTEGRATION: XMLBLASTER, INFOPACKS, AND COMPONENTS

InformANTS models user interest and relevant documents in a common format to support its main function: bringing users and documents together for enhanced information retrieval of dynamic information and to foster collaboration between users with similar interest. We call this common format (an InformANTS-internal data structure) the InfoPack.

Architecturally, an InfoPack consists of a set of meta-level attributes that describe, for instance, its origin (a particular user ID or the URL of a document) or the level of attention that the system should give it. The InfoPack also carries a list of object model instances, each of which carries model data and an identifier of the specific object modeler type that generated it. As we will refine below, the NIST evaluation prototype supports five different object model types, which may or may not be present in a particular session.

Any given InfoPack relates to a specific entity outside InformANTS. Each user model InfoPack corresponds to a particular analyst that has been or is currently working with the InformANTS system. Each document model InfoPack contains the object models for a particular document accessible in the corpus. When the contents of a document changes (as it is for instance the case with Wiki pages) or as the user interest evolves, the object models in the corresponding InfoPacks are updated. If new documents or users "arrive" new InfoPacks are created for them.

InformANTS has the following main components:

- IOMS (Information Object Modeling Service), managing a number of Object Modeler instances
- UMS (User Modeling System)
- IMS (Information Matching System), comprising a number of Object Model Reasoner instances and an InfoPack Locator instance

The IOMS consumes the stream of analytic events (ALE) from the analytic event server (ALS), models any associated contents with one or more object modeler instance, and announces the ALE's and the associated content models to the UMS. Furthermore it provides the modeling service for document contents and releases these models as (document) InfoPacks back into the system.

The UMS consumes the stream of ALE's and the object models of their associated contents and creates or updates models of the user interest which are announced to the system as (user) InfoPacks.

The IMS organizes the current set of user and document models by similarity in the InformANTS information space and extracts relevance orderings of documents or users relative to a given user or document model.

2.1.1.3 INFORMANTS WEBSERVICES

The InformANTS system provides a number of webservices: to support recommendations of relevant documents and users for other users or documents, to extract Virtual Interest Group (VIG) recommendations from the UMS, to generate keyword queries from user models, and to download, model, and re-rank a list of documents relative to the interest of a given user. The latter two (query generation and re-rank) were used in the final NIST evaluation. Table 1 lists the relevant calls and their parameters and provides a short description.

Table 1. InformANTS Webservice Methods.

Method	Parameters	Description
GetInfo Pack	ID	Returns the full InfoPack with the given **ID**
GetUser Neighbors	ID, maxCnt	Returns a list of InfoPacks up to length **maxCnt** that are a) document InfoPacks, and b) near the InfoPack with the given **ID** in the IMS information space
Get Document Neighbors	ID, maxCnt	Returns a list of InfoPacks up to length **maxCnt** that are a) user model InfoPacks, and b) near the InfoPack with the given **ID** in the IMS information space
GetQuery FromUser Model	ID, maxCnt	Returns a keyword query string with up to **maxCnt** terms generated from the user model with the given **ID**
Get ReRanked Query	ID, maxCnt, List<URL, text>	Returns the top **maxCnt** elements of a re-ordering of the elements of the **List** based on either the contents of the **URL** or the associated **text** relative to the user InfoPack with the given **ID**
GetUmsVig	ID	Returns a list of other user models ordered by similarity by the UMS relative to the user model with the given **ID**

2.1.1.4 INFORMANTS-INTERNAL PROTOCOLS

The integration of the various components of the InformANTS system via the xmlBlaster communications infrastructure requires a number of interaction protocols. In the following, we discuss the two most important protocols in detail.

2.1.1.4.1 P1: FROM ALE TO USER AND DOC MODELS

Protocol 1 (Figure 4) is triggered by the arrival of a new ALE in the ALSClient of the IOMS component. If the ALE carries a URL reference to contents associated with this user action, the IOMS downloads the contents otherwise the text snippet in the ALE are used directly. The raw contents are prepared (de-formatting, term-frequency analysis) by the IOMS and then announced to the available object modeler instances. Each object modeler that receives this announcement processes the prepared contents with its specific modeling technique and returns an object model instance to the IOMS.

The IOMS, knowing which object modeler instances are currently active, waits for all object models to return for a given ALE and then announces the ALE and the associated object models to the UMS. The ALE's are released by the IOMS in sequence of the timestamps of the user actions. Also, if the contents associated with the ALE pointed to an external document (URL in ALE), then the IOMS constructs a document InfoPack from the object models and announces it to the IMS (Locator, Reasoners, DB).

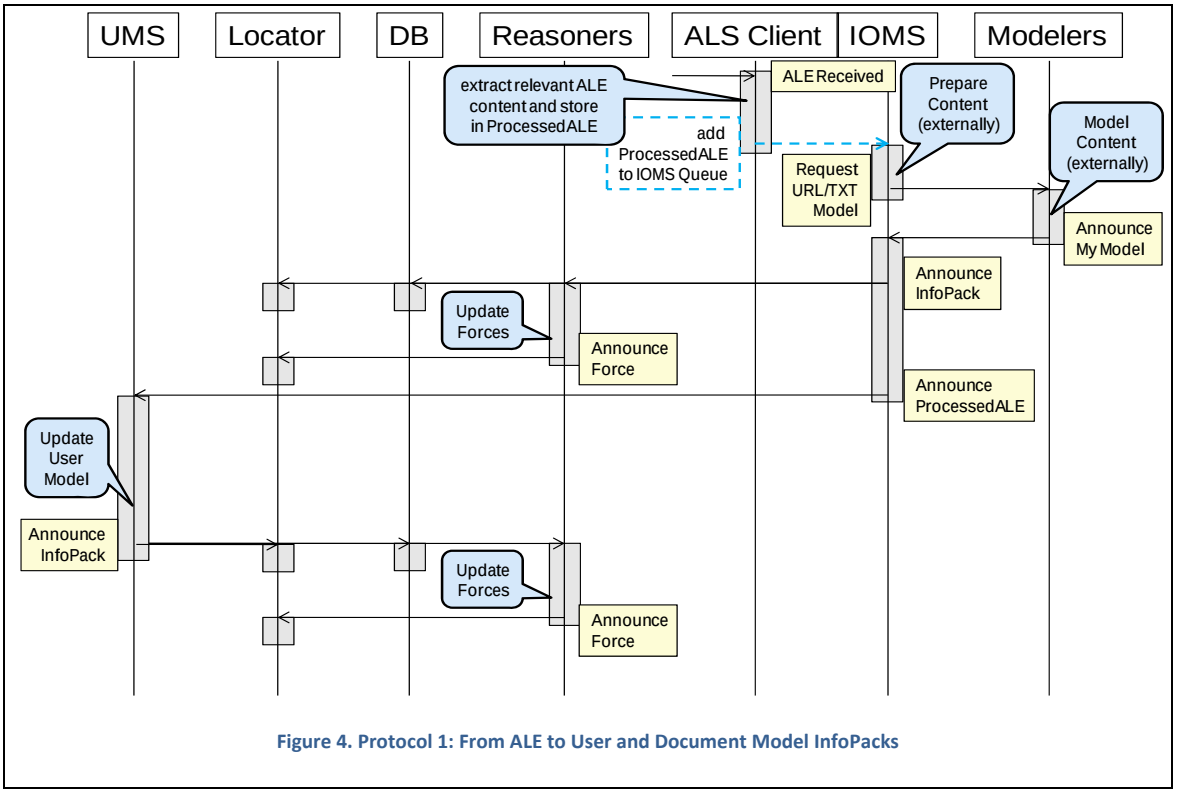

Figure 4. Protocol 1: From ALE to User and Document Model InfoPacks

The UMS receives announcements of ALE's and their object models and updates the model of the user that triggered the ALE accordingly. This user model update results in an announcement of a new or updated user InfoPack to the IMS.

Any announced InfoPacks (document InfoPack from IOMS or user InfoPack from UMS) will be processed by the object model reasoners, who determine a set of forces (see section 2.1.4.1) for the InfoPack and communicate those to the InfoPack locator component. The locator and the database also process the InfoPack announcements to maintain their up-to-date representations of the set of currently known InfoPacks.

2.1.1.4.2 P3: WEBSERVICE RE-RANKING CALL

Protocol 3 (Figure 5) is triggered by a GetReRankedQuery webservice method request (see Table 1). The request is passed on to the InfoPack locator component for processing and the response is only to be returned by the webservice once the results from the locator arrive.

The InfoPack locator first iterates over all elements in the list of document references provided with the request and determine which documents already correspond to a document InfoPack in the system. These InfoPacks may exist either because the user already viewed the document according to a previous ALE (see Protocol 1), or because it was modeled for a previous re-rank request already. For all other documents that have no corresponding InfoPack, the locator issues a modeling request to the IOMS.

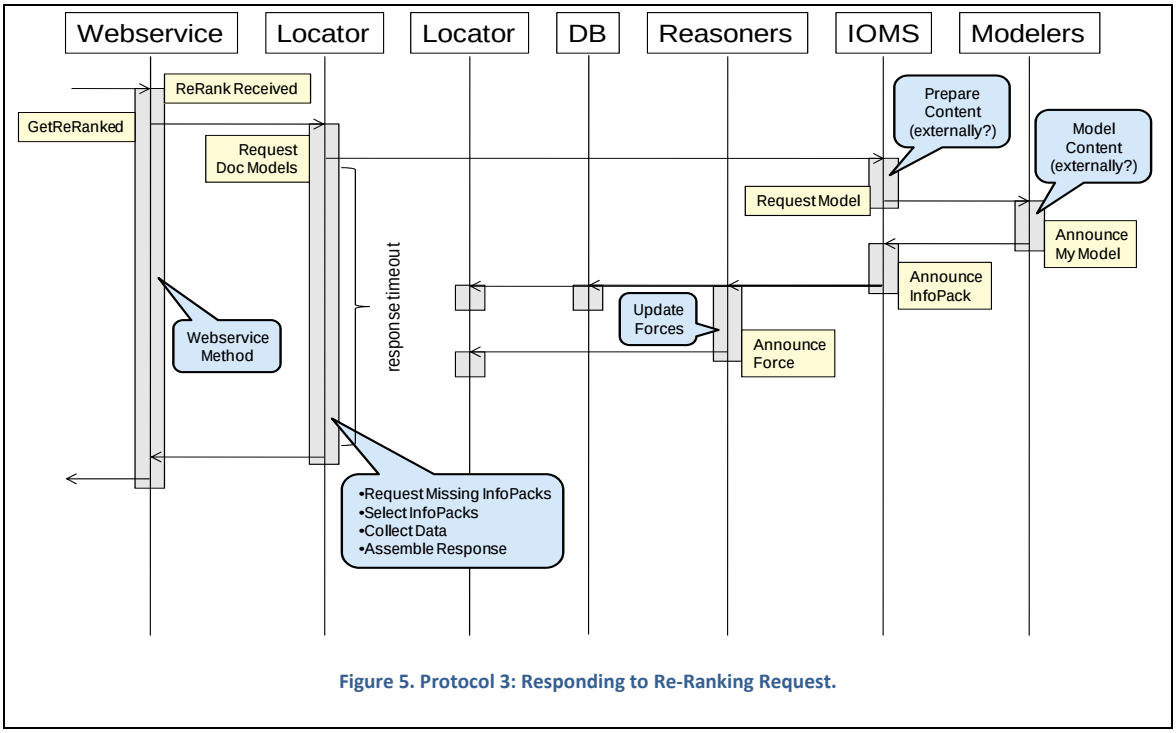

Figure 5. Protocol 3: Responding to Re-Ranking Request.

The IOMS downloads the contents for each requested document, prepares the raw contents for modeling (de-formatting, term-frequency analysis), and announces the prepared contents to the currently active object modeler instances. Each object modeler that receives these announcements processes the prepared contents with its specific modeling technique and returns an object model instance to the IOMS.

The IOMS, knowing which object modeler instances are currently active, waits for all object models to return for a given document and then creates and announces a document InfoPack with the associated object models and the document meta information (URL). Any announced InfoPacks are processed by the object model reasoners, which determine a set of forces (see section 2.1.4.1) for the InfoPack and communicate those to the InfoPack locator component. The locator and the database also process the InfoPack announcements to maintain their up-to-date representations of the set of currently known InfoPacks.

After requesting (InfoPack) models for the missing documents, the locator pauses the processing of the re-rank request for a fixed amount of time that is configured in the InformANTS setup (110 seconds in the NIST evaluation). This pause is intended to give the IOMS time to download and model any missing documents and the locator to update the organization of the user and document InfoPacks in its information space. Once the required delay has passed, the locator determines the new ordering of documents from the request list by their distance to the identified user InfoPack in the information space. It returns the requested top N documents in this new ordering to the webservice component, which hands it off to the caller of the webservice method.

2.1.2 INFORMATION OBJECT MODELING SYSTEM (IOMS)

The Information Object Modeling System (IOMS) has two main functions. First, it is the entry point of the flow of Analyst Logging Events (ALE) into the InformANTS system, which provides the information necessary to model the evolving user interest. Second, it uses one or more modeling techniques to turn raw (textual) contents into structured models of this content.

The first function is provided by the ALSClient component of the IOMS (section 2.1.2.1), which operates in an independent thread within the IOMS. The second function is supported by a set on object modeler classes (section 2.1.2.3) that may be hosted inside the IOMS process or run on separate hosts to "harvest" more computational power for the InformANTS system. The IOMS downloads (if necessary) and prepares the content that is processed by these modelers (section 2.1.2.2).

2.1.2.1 IOMS COMPONENT: ALSCLIENT

The ALSClient is InformANTS tool to access Analyst Log Events (ALEs) stored in the ALS. The ALEs retrieved are processed and pushed through the system, triggering user modeling, document modeling, VIG modeling, and, depending on set up, population of documents in the Locator.

When InformANTS is running to support live analysts, the ALSClient queries the ALS at regular intervals for recently logged user events. (It does so thoroughly, internally keeping tally of all observed ALEs and making overlapping queries.) In addition to providing the connection to the ALS when the system is live, the ALSClient also supports querying broad ranges of time to support experiments or to initialize InformANTS with a state reflecting the accumulate behavior of analysts over a period of time before system initialization.

Upon receipt of ALEs, the ALSClient processes each one, extracting meta-data and relevant content according to the ALE spec. The metadata as well as text or document URL are placed in an object (Processed ALE, or 'PALE') used as a common representation shared among several of the major system components.

2.1.2.2 IOMS COMPONENT: CONTENTS PREPARATION AND DCS

The contents that the IOMS is asked to model is either provided directly (e.g., snippet of analyst-copied text in an ALE) or by URL reference (e.g., Google-recommended document in re-rank request, link to an analyst-viewed page in an ALE). If the contents is provided by reference, then the IOMS uses an Oculus-provided webservice (Document Conversion Service, DCS) to download the contents from the specified URL and to remove (as much as possible) any document formatting markup (e.g., HTML tags). To avoid overload of the InformANTS system, the DCS-webservice method permits to limit the size of the returned raw text. In the NIST evaluation experiment, we set this parameter to 20k bytes.

Once the raw content is available, it can be handed off to the Object Modelers.

2.1.2.3 IOMS COMPONENT: OBJECT MODELERS

The primary purpose of the IOMS is to model raw contents using one or more modeling techniques. Each modeling technique is implemented by a different object modeler class, but all such classes provide the same API to the IOMS. For a particular session, InformANTS can be configured to instantiate a given set (one or more) object modeler classes. In preparation of the NIST evaluation, we implemented five different modeling approaches in collaboration with other CASE teams:

- Text to Specialized Concept Maps (T2SCM, by SET)
- Term Frequency (TF, by NewVectors)
- Text Analysis Engine (TAE, by FairIsaac)
- Latent Dirichlet Allocation (LDA, by BAE)
- Pachinco Allocation Model (PAM, by BAE)

The T2SCM modeler consumes the raw content obtained in the previous document preparation step. All the other modelers require that the content first be transformed into a term-frequency representation. Our colleagues from BAE kindly provided a library that performed this transformation for the IOMS.

Using the xmlBlaster infrastructure, the IOMS announces a modeling request (carrying the raw content or the term-frequency representation) to all instantiated object modeler processes. Then the IOMS waits for a given time for a response from the modelers, before it proceeds with the announcement of the ALE to the UMS or document InfoPacks to the IMS.

In the following, we discuss the modeling approach for each of the five object modeler instances.

2.1.2.3.1 IOMS COMPONENT: OBJECT MODELERS – T2SCM

The T2SCM (Text to Specialized Concept Map) tool extracts typed concepts and relations from English text and generates a specialized concept map represented as an InfoPack XML document. It is based on the open source NLP software GATE (General Architecture for Text Engineering)[1] and has no relation to the AntCAFÉ T2CM tool developed in the NIMD program. The architecture for the tool is shown in Figure 6. The input text is first processed by GATE to extract named entities or annotations. The entities are then processed to derive typed concepts and proximity-based relations. The concepts and relations are formatted to form an InfoPack XML document.

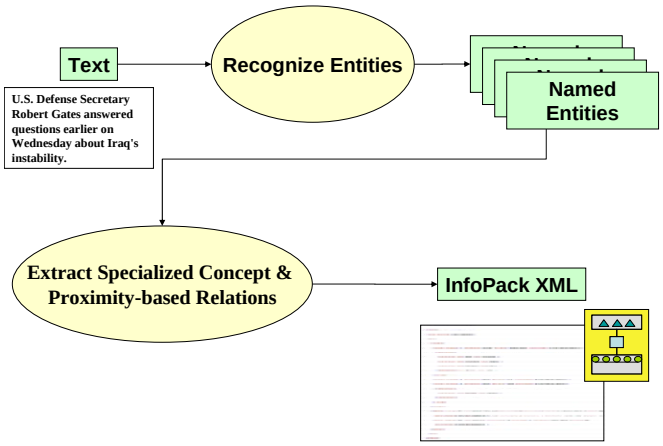

Figure 6. T2SCM architecture.

GATE can annotate Date, FirstPerson (first names), Identifier (ID's), JobTitle, Location, Lookup (anything that appears in the gazzetteer), Money, Organization, Percent, Person, Sentence, SpaceToken, Split (punctuation that marks end of sentences), Temp (things marked as potentially important but not categorized), Title, Token, Unknown, and DEFAULT_TOKEN (punctuation like quotes, apostrophe's, etc). In addition, GATE allows the custom creation of annotation types, e.g. we can create a "Terror" annotation and put things like "terrorism" "terror" and "terrorist" etc. in it. Based on the annotations, T2SCM creates concept types including persons and places. For person concepts, T2SCM extracts job title, first name, last name, and gender as characteristics. For location concepts, the location type (e.g. country or city) information is provided as a characteristic.

[1] http://gate.ac.uk/

The T2SCM tool is enriched with external sources. For persons, it looks up the nndb Website (www.nndb.com) to find date of birth, state of being alive or dead, gender, race or ethnicity, and occupation. For places, the tool looks up the CIA fact book (https://www.cia.gov/library/publications/the-world-factbook/) and the airport database (http://www.partow.net/miscellaneous/airportdatabase/index.html#GlobalAirportDatabaseLicense) for the latitude and longitude of the location. In addition, each relation contains a characteristic "co-occurrenceWeight" to indicate the strength of co-occurrence of the relation. The default value is 1.0, 0.5, and 0.1 for sentences-, paragraph-, and document-level relation, respectively. Furthermore, a Web service for the T2SCM tool has been created and awaits deployment.

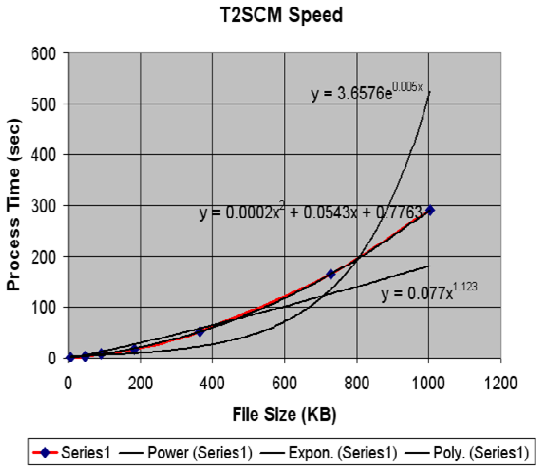

Figure 7. T2SCM speed.

The weight of a concept indicates its importance within the document, not to an analyst. The weight is determined as follows:

$$W = Tc + Bc * MIN(COUNT/SQRT(WORDS), 1)$$

The variable COUNT is the number of occurrences of the concept and WORDS is the total number of words in the document. The resultant range for concept weight is [0.3, 0.6] assuming constant Tc = 0.3 and Bc = 0.3. Both Tc and Bc can be adjusted via a configuration file.

T2SCM extracts three types of proximity-based relations:

- Sentence-wise Relation (SR) – two constituent concepts are occurring in the same sentences.
- Paragraph-wise Relation (PR) – two constituent concepts are occurring in the same paragraph.
- Document-wise Relation (DR) – two constituent concepts are occurring in the same document.

Note: formula values are tentative parameter defaults.

Relation weight is determined as follows:

$$W = Tr + Br * MIN(COUNT/SQRT(WORDS), 1)$$

The variable COUNT is the number of occurrences of the relation and WORDS is the total number of words in the document. The resultant range for concept weight is [0.3, 0.6], assume that constant Br = 0.1, and Tr's values for DR, PR, and SR are 0.3, 0.4, and 0.5, respectively. Both Tr and Br can be adjusted via a configuration file.

The speed of T2SCM is shown in Figure 7. It is an order 2 polynomial function of input file size (in bytes). When the file is under 200 KB, the speed is close to linear. A 92 KB text document (approximately 15,104 words) takes 8 seconds to process on a Windows XP machine (Dell Latitude D620, Genuine Intel CPU T2400 @ 1.83GHz 987 MHz, 1.00 GB of RAM).

Figure 8 shows 3 screenshots from the T2SCM demo. The top screenshot shows the input text, which is the following sentence:

"U.S. Defense Secretary Robert Gates answered questions earlier on Wednesday about Iraq's instability."

The pane with the XML View tab displays the generated InfoPack XML string. The middle screenshot shows the tree view of the concepts portion of the generated concept map. The bottom screenshot shows the relation portion.

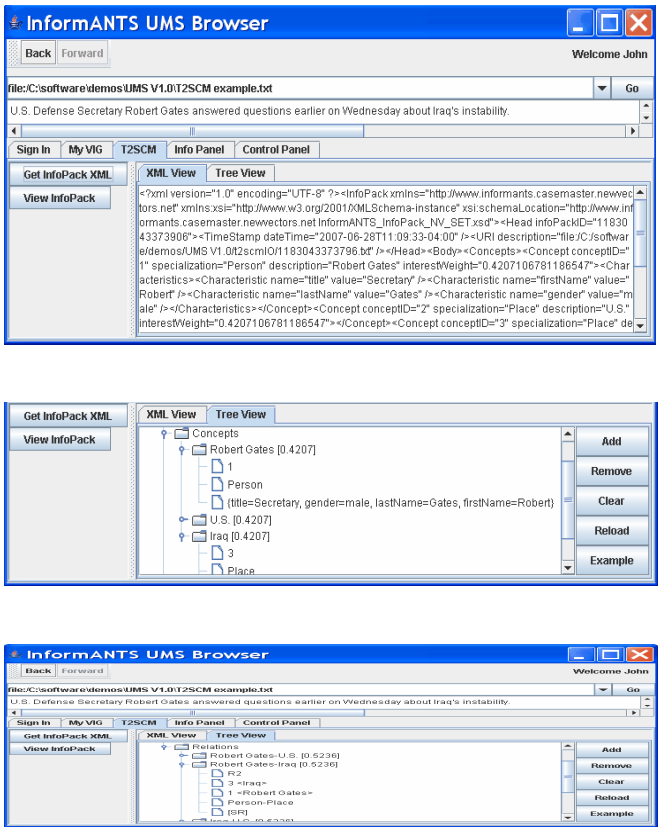

Figure 8. T2SCM demo screenshots

2.1.2.3.2 IOMS COMPONENT: OBJECT MODELERS – TF

The IOMS component that handles document downloading and preparation hands to the topic modelers raw text that has been preprocessed into a filtered list of terms and their frequencies in the document. The topic object modelers, which the Term-Frequency (TF) modeler can notionally be considered to be, convert the list of term frequencies into a document object model that can be ingested by the topic reasoners which will be described later. The document object model itself is a discrete normalized probability distribution over a set of unique markers. The set of markers for each type of topic modeler is specific to the model type and is determined before system start up. The markers in each set are chosen, according to the model type, to be an efficient and complete representation of the abstract space of documents. Each document model, thus, is represented by a single point in this abstract space.

During the system initialization phase, each topic modeler is given its own Information Space Model (ISM) file. The ISM files were generated a priori from a large (65,000) corpus of training documents. The techniques used to generate the ISM file for each topic model type, as well as the methods used to generate the document model from the document's term frequencies, are described in this and in the following sections.

The ISM file for the TF modeler contains a single long list of words which is intended to represent the most common words in the corpus of training documents. This list of words constitutes the marker set for the TF modeler. While the markers in the TF model represent the most common words in the training documents, the list was not generated directly from the corpus. It was generated from two of the other modelers' ISM files, which were generated directly from the corpus. The TF ISM file that was used predominantly in the NIST evaluation experiments contained the union of the top 5,000 most frequent words from each of the ISM files from the TAE and the LDA modelers. This set contained a total of 6,503 words.

The document model is simply the normalized term frequencies of those terms in the document that are present in the marker set. All markers in the marker set that are not present in the document are considered to have a weight of zero.

2.1.2.3.3 IOMS COMPONENT: OBJECT MODELERS – TAE

The TAE, LDA, and PAM object modelers are, in a more strict sense, considered to be topic modelers. The marker set for the true topic modelers is made up of topics, or groups of words, rather than individual words (as is the case with TF.) The ISM file for each topic modeler specifies which words are in which topics, how the words are weighted within the topic, and how the topics are weighted within the training corpus from which the model was generated.

The TAE modeler is based on an algorithm[2] proprietary to FairIsaac. The algorithm takes every unique word found in the training documents and partitions them among a pre-determined number of topics. The number of topics used in the NIST evaluation experiments was 200.

The topic modeler generates the document model from the list of term frequencies passed in for each document. Because the document model is considered a probability distribution over markers, which in this case are topics rather than words, some manipulation of the document's term frequencies is required. In Equation 1, which describes the calculation for the probability of topic i in document D, p(w|D) represents the term frequencies for the document. The other terms are pulled directly from the information provided in the ISM file. The sum is over all words in the training corpus. $p(w|t_i)$ is the probability of word w in topic i. $p(t_i)$ is the marginal probability of topic i. p(w) is the marginal probability of word w in the training corpus.

$$p(t_i \mid D) = \sum_{w \in W} \frac{p(w \mid t_i)\, p(t_i)}{p(w)}\, p(w \mid D)$$

Equation 1. Construction of the TAE document model.

[2] A Workbench for Rapid Development of Extraction Capabilities. Dayne Freitag, Matthias Blume, John Byrnes, Richard Calmbach, and Richard Rohwer. 2005 International Conference on Intelligence Analysis.

2.1.2.3.4 IOMS COMPONENT: OBJECT MODELERS – LDA

The ISM file for the Latent Dirichlet Allocation (LDA)[3] topic modeler was generated by BAE. The number of topics used in the NIST evaluation experiments was 200.

The process for converting a document's list of term frequencies into a document model for the LDA topic modeler is exactly the same as for the TAE modeler. The weight of each topic in the corpus of training documents was not provided in the LDA ISM file, so topics were assumed to be uniformly distributed.

2.1.2.3.5 IOMS COMPONENT: OBJECT MODELERS – PAM

The ISM file for the Pachinco Allocation Model (PAM)[4] topic modeler was generated by BAE Systems. The number of topics used in the NIST evaluation experiments was 200.

The process for converting a document's list of term frequencies into a document model for the PAM topic modeler is exactly the same as for the TAE and LDA modelers. The weight of each topic in the corpus of training documents was not provided in the PAM ISM file, so topics were assumed to be uniformly distributed.

2.1.3 USER MODELING SYSTEM (UMS)

In InformANTS, one key role of user modeling is to build models of intelligence producers or consumers. The user model captures the user's interests and preferences. A user model is represented as an information packet (InfoPack) whose body consists of specialized concept maps. These maps contain five classes of concepts: person, place, action, object, and a general catch-all class. The maps may also contain general relations among the concepts. Both the concepts and relations carry weights that indicate the user's level of interest.

The models we build are initially based on a description of each user's tasking (in natural language). They are further enhanced by extracting terms from ongoing user queries, assertions and relevant documents, and they can be further refined based on an underlying ontology such as WordNet[12, 13]. The weights of the concepts and relations are dynamically adapted via either implicit or explicit feedback. More precisely, we employ a type of reinforcement learning, where the current model is used to predict future user actions (e.g., which web page the user will select), and when future actions diverge from the predictions (a different URL is selected), an adaptation cycle is invoked.

We refer to the use of a user model to shape the behavior of an information system as model-guided processing. We have demonstrated this approach in previous work. For example, in a previous effort for the Defense Logistics

---

[3] Topical N-grams: Phrase and Topic Discovery, with an Application to Information Retrieval. Xuerui Wang, Andrew McCallum and Xing Wei, Proceedings of the 7th IEEE International Conference on Data Mining (**ICDM**), 2007. http://www.cs.umass.edu/~mccallum/papers/tng-icdm07.pdf

[4] Mixtures of Hierarchical Topics with Pachinko Allocation. David Mimno, Wei Li and Andrew McCallum. **ICML**, 2007. http://www.cs.umass.edu/~mccallum/papers/hpam-icml2007.pdf

Agency we demonstrated how to personalize a search for obsolete replacement parts [12]. As performers in DTO NIMD, we obtained very encouraging results by using our models to guide a swarm-based text search mechanism [14]. In DTO VACE (Phase II), we demonstrated how to use models to alert a ground-camera security system to anomalous behavior.

In this section, we first analyze user relevance feedback and how it relates to user modeling. We then describe two modeling algorithms with the first based on contextual feedback and the second on more general feedback types. We end this section by describing some tuning experiments we have performed.

2.1.3.1 UMS INPUT: USER RELEVANCE FEEDBACK

When interacting with an information system, the user performs various actions (e.g. reading, and cutting/pasting) on various targets (document and passage). User event refers to instance of a user performing an action on an information object. User's interests and preferences are often embodied in user events. For example, if a user queries about "user modeling", we can presume that the user has an interest on this topic. When user events reflect user's interests, they become relevance feedback. User modeling techniques rely on relevance feedback to capture user's interests and represent them explicitly.

2.1.3.1.1 USER EVENT DECOMPOSITION

A user event can be decomposed into six components: *subject, action, object, time, task, and application* (Figure 9). Subject, or actor, refers to the originator of the event, which is often the current user. Action is the specific deed that the user has performed, e.g. reading, saving, printing, cutting/pasting. Action is considered the relevance predictor for the event. Different actions are unequal in their potential for inferring user's interests, e.g. The cutting/pasting action reflects more of user's interest than the browsing action. Object is the information target of the action, e.g. a document is the object of a reading action. This is where the information about user's interests is contained.

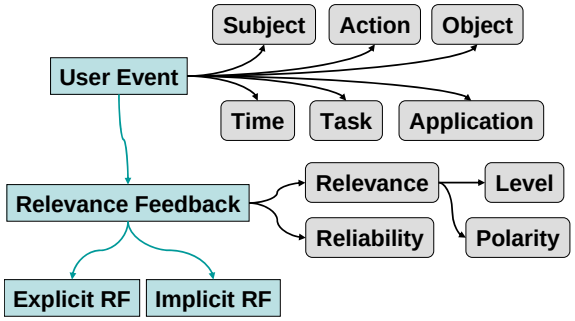

Figure 9. User Event Decomposition.

The time component refers to the time when the event was generated. More recent events are more indicative of user's interests than older events. It is thus often used by user modeling systems to decrease the influence of older events. The task and application component refer to the assignment and tool the user is executing when the event is generated. They provide the context for the user event, which modeling systems may find useful when interpreting the event.

2.1.3.1.2 USER EVENT TAXONOMY

There are many types of user events that might occur in information systems. Clearly, there are multiple ways to organize them. For example, they can be divided into simple vs. complex events. A composite event is a high level event that consists of simple events. A simple event is a low level event that cannot be decomposed any further. A complex reading event consists of simple events including: opening a document, scrolling the pages, pausing, and closing the document.

NIST and Oculus Info, Inc. are leading an effort under DTO CASE program to develop a taxonomy for analytic events, which are user events generated by intelligence analysts.

2.1.3.1.3 RELEVANCE FEEDBACK

Relevance feedback is a user event that is both reliable and relevant. Reliability refers to the consistency with which the event is reflecting user's current interests. Relevance refers to the extent that the event is reflecting user's current interests.

2.1.3.1.3.1 EXPLICIT RELEVANCE FEEDBACK

With an explicit relevance feedback, the level of relevance of the event is directly stated by the user. For examples, rating and voting belong to this type of feedback. This type of user feedback is ideal for user modeling because it reduces the effort in inferring the user's interest. The flip side is that it requires an extra effort on the part of the user to express the relevance of the event, which may or may not be acceptable depending on the application.

2.1.3.1.3.2 IMPLICIT RELEVANCE FEEDBACK

With an implicit relevance feedback, the level of relevance of the event is inferred by the information system. Example feedback events include reading, query, cut/paste, and surfing a Web page. This type of feedback is unobtrusive to the user but put the burden on the user modeling system to figure out the user relevance.

2.1.3.1.4 USER RELEVANCE

User relevance of an event denotes the correspondence between the information in the object of the event and the user's true interests. This correspondence is determined by the action of the event. E.g., a query event is more relevant than a reading event. The cut/paste text is more relevant than the rest of the text in the same document.

User's interests can be positive or negative, e.g. "like reading" is positive, while "dislike shopping" is negative. Both can be part of a user's interest profile.

Table 2. Relevance assignments for user events.

User Event[5]	*Relevance Level*	*Polarity*
Query (Keywords/Question/Assertion)	1	+1
cut/paste	0.9	+1
Selection from list	0.8	+1
Saving/printing	0.7	+1
Chat	0.6	+1
Reading doc/Web link activation/ Opening doc	0.5	+1
Non-Selection from list	0.8	-1
Deletion	1	-1

2.1.3.1.4.1 INFERRED

We recognize that different types of feedback events reflect user's interests to a different degree. In other words, they have different level of relevance to user's interests. For example, a block of text appeared in a query is more relevant than the same text in a document that the user read about. We propose a tentative scheme of relevance level assignments for common types of feedback events (Table 2).

2.1.3.1.4.2 STATED

In the case that user ratings are available, the rating is the relevance of feedback event. The rating scale needs to be reconciled with the relevance scheme shown in Table 2.

2.1.3.1.4.3 POLARITY

The polarity of a feedback event indicates whether it positively or negatively reflects user's interests (Table 2). A positive polarity means that the event is relevant to the user's interests whereas a negative polarity means that the event is not relevant to the user's interests. A user event can also be neutral, i.e. providing no information on user's interests. In that case, the polarity takes the value of zero.

Positive polarity examples: Query, Question/Assertion, cut/paste, chat, reading, saving, printing, and positive rating.

Negative polarity examples: deletion, negative rating, and non-selection.

2.1.3.1.4.4 RELATIVITY OF RELEVANCE

Absolute relevance – It may be argued that there exists some absolute relevance. For example, a block of text seems always more relevant if it is part of a cutting/pasting event than if it appears in a reading event. However, more often than not, relevance depends on the user context. For example, a document selected from a set of documents implies that this document is more relevant than the rest in that set. It does not say anything about its relevance compared to documents outside of that set.

[5] This is referred to as analytical event in the context of intelligence analysis.

Relevance also depends on the user. In other words, different user employs events differently and the extent a particular event reflects the user's interest varies from user to user. Furthermore, relevance may depend on the task and application, i.e. the same user may employ events differently under different tasks or in different applications (e.g. Web surfing vs. emailing).

The relative state of the relevance of a user event behooves us that we should be careful in assigning the relevance levels in order to infer user's interests. These assignments are approximations and may contain errors. Consequently, assignments themselves should be programmed into systems with flexibility such that they can be adjusted depending on the application settings.

2.1.3.2 MODELING ALOGORITHM I: BUBBLE UP BASED MODEL ADAPTATION (BUMA)

BUMA combines *Merging algorithms*, methods for content adaptation as described above, and *Extended Bubble Up* [1], a weight adaptation algorithm we developed previously under NIMD program. The *Extended Bubble Up* algorithm requires ***contextual feedback***, a relevance feedback in the form of a user choice made from a user context. If a user selected one relevant document from a list of documents, then the selection is the user choice whereas the list of documents is the user context. A query can also be framed as a contextual feedback as follows. We take the current query as the user choice and N previous queries as the user context. With internal testing, we found that such feedback achieves the effect of *aging*, which is the decreasing of importance of concepts or relations over time without reinforcement. Note the N should be small, and preferably less or equal to the magic number 7 [2]. The intuition there is that when a user thinks about a new query, roughly seven previous queries are probably still lingering on user's mind and are being intuitively compared. As a result, this small set of previous queries serves as the context for the new query.

Given a contextual feedback, the pseudo-code for BUMA goes as follows:

1) Using NLP tools to extract CR's[6] from the feedback
2) Adapt the weights of the current model with the feedback using Extended Bubble Up.
3) Insert into the current model the top N new CR's from the feedback with a default weight modulated by their relevancy (e.g. term frequency).

BUMA can be applied to the *front page news prediction* problem with the Factiva data (see section 4.3). In terms of the relevance feedback, the actual front page story is the user (i.e. the newspaper editor) choice whereas other articles on other pages of the same publication are from the user context.

2.1.3.2.1 DEFAULT WEIGHTING OF NEW CR'S

If the extracted CR does not exist in the current user model, the new CR is inserted into the model with a default weight. The default weight assignment is proportional to the *relevance* of the feedback event (see **Table 2**). In addition, the default weight also takes into account the significance (e.g. term frequency) of the CR in its originating feedback. The default weight W_i for ith CR of feedback f is computed as follows.

[6] The acronym "CR" stands for "concept or relation".

$$W_i = R_f W_m + B\frac{S_i}{S_{\max}}, \text{ where}$$

W*m* is the ***maximal initial weight*** across all types of feedback, R_f is the relevance for feedback *f*, S_i is the significance value of the ith CR, S_{max} is the maximal significance value in feedback *f*, B is ***the boosting factor*** with value range of [0, 0.1] and default value of 0.01.

2.1.3.3 MODELING ALGORITHM II: REINFORCEMENT AND AGING BASED MODEL ADAPTATION (RAMA)

Given a relevance feedback, the pseudo-code for the RAMA algorithm goes as follows:

1) Using NLP tools to extract CRs from the feedback.

2) Age the current model by applying a forgetting function to all CRs in current model.

3) If a CR from the feedback already exists in the model, its weight in the model is positively or negatively reinforced. The resultant weight is further changed to the default weight if and only if the value is less than the default weight and *the reinforcement is positive*.

3) Insert top N new CRs from the feedback into the current model with a default weight modulated by their relevancy (e.g. term frequency).

2.1.3.3.1 INITIAL WEIGHTING OF NEW CR'S

See section 2.1.3.2.1.

2.1.3.3.2 REINFORCEMENT OF ACTIVATED CR'S

An activated CR refers to a CR that exists in both the user model and the feedback event. It triggers a positive or negative reinforcement depending on the polarity of the feedback event. If the polarity is neutral, then the feedback event is not useful and is ignored.

2.1.3.3.2.1 POSITIVE REINFORCEMENT

Positive reinforcement occurs when the polarity of the feedback is positive. The activated CR is reinforced via an increase in its weight.

$$W' = Max(W + R_f I(1-W), R_f W_m), \text{ where}$$

W' is the new weight, W is the current weight, R_f is the relevance for feedback f, and ***I*** is ***the reinforcement factor***, which determines the size of the weight increase. The value of *I* falls in the range of [0, 1]. The larger the value for I is, the larger the increase. The default value for I is 0.01. Note that the second term ($R_f W_m$) in the Max function is the default weight for the current feedback.

2.1.3.3.2.2 NEGATIVE REINFORCEMENT

Positive reinforcement occurs when the polarity of the feedback is negative. The activated CR is reinforced via a decrease in its weight as follows:

$$W' = W - R_f I W$$

The symbols have the same meanings as in previous section. Note the larger the value of the reinforcement factor, the larger the decrease.

2.1.3.3.3 AGING OF NON-ACTIVATED CR'S

Forgetting function is applied to CR's that are not reinforced by the feedback event.

2.1.3.3.3.1 AGING FREQUENCY

Triggered by Feedback Event - Forgetting function gets applied when a feedback event occurs.

Periodically Triggered - Forgetting function gets applied when a specific interval has expired.

2.1.3.3.3.2 AGING RATE

We decrease the weight as a linear function of current weight:

$$W' = (1 - A)W$$

A is the ***aging factor*** with value in the range of [0, 1]. The default value is 0.001. Note this value is one tenth of the value of the *reinforcement factor*, which means the rate of forgetting is much slower than the rate of reinforcement. We can also think of A as the evaporation rate in the pheromone analogy.

2.1.3.3.3.3 USER MODELING DEMO

This demo is built upon RAMA adaptation algorithm. The architecture is shown in Figure 10. The protocol for running the demo is as follows:

- A user logs in to UMS
- User visits pages on Wikipedia or other Web sites
- UMS generates interest model for the user based on visited pages using RAMA algorithm
- User views the model

Figure 11 shows three screenshots from the user modeling demo. The user "John" signed in and issued a query "Iraq"

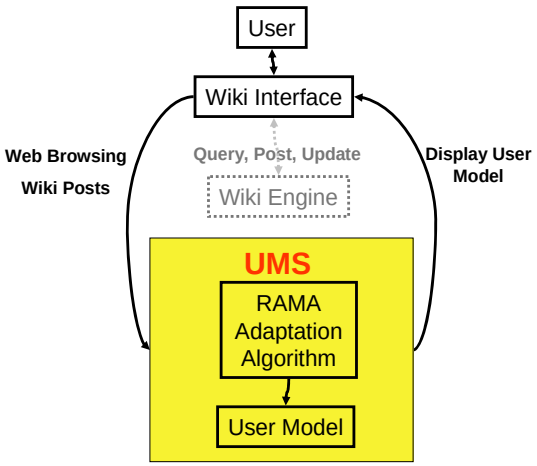

Figure 10. User modeling demo architecture.

at the Wikipedia web site (left). The user then browsed the Iraq page (middle). She continued to surf the "Baghdad" page and clicked the "My Model" button to view the current user model (right). The user model captured concepts like Iraq, Baghdad, Saddam Hussein, and Ottoman Empire. It also captured proximity-based relations such as "Iraq – Baghdad" and "Saddam Hussein – Iraq" (not visible in the right screenshot).

2.1.3.4 UMS TUNING VIA SIMULATION

We have evaluated the adaptation algorithm RAMA with two approaches: 1) Feeding repeated Analytic Logging Events (ALEs); and 2) Simulated User. The latter approach contains two separate experiments, one with synthetic data and the other with open Web data.

2.1.3.4.1 FEEDING REPEATED ALES

With this approach, we auto-generate repeated ALEs of various types, feed them into the RAMA algorithm, and then observe the change in the weight of the concept in the evolving user model. Note, the user model is initially empty. An example of ALE sequences is shown below:

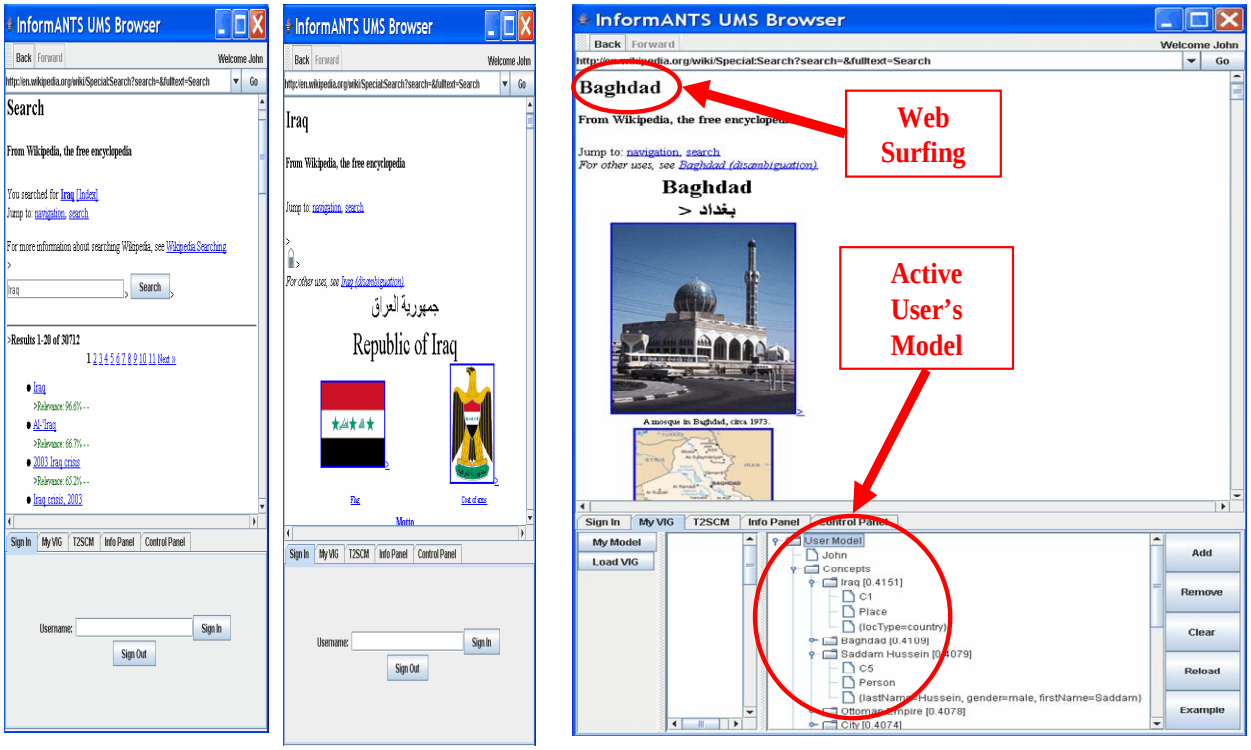

Figure 11. User modeling demo screenshots.

u1,search,Iraq,100

u1,access,Iran,100

u1,retain,Japan,100

u1,access,Iran and Japan,100

The first line reads, "User u1 searched the concept Iraq 100 times." The results are shown in Figure 12. The 3 panels have different RAMA parameter settings. The top one has the reinforcement factor (I) at 0.25 and the aging factor (A) at 0.005. For the middle panel, the settings are I=0.25, and A=0.01. For the bottom panel, I=0.5, and A=0.01.

In the top panel, the blue trace is for the concept "Iraq". With 100 search events on this concept, its weight starts at 0.7 and increases quickly initially to about 1.0 and stays there until the 100th event. The weight then gradually decline beyond the 100th ALE event because this concept is no longer being reinforced. The next 100 ALEs are accesses to the concept "Iran", we see a similar pattern except at the tail where it rises quickly and then levels off (the magenta trace). The pattern at the tail is caused by the 100 access ALEs on the relation "Iran-Japan".

The middle panel differs from the top panel in that its Aging factor value doubles that of the top panel. We observe that the weight drops quickly when a concept is not reinforced via appropriate ALEs.

The bottom panel has a higher reinforcement factor value than that of the middle panel. By comparing the corresponding traces, we see that the weight rises faster and plateaus at higher level in the bottom panel than the middle panel.

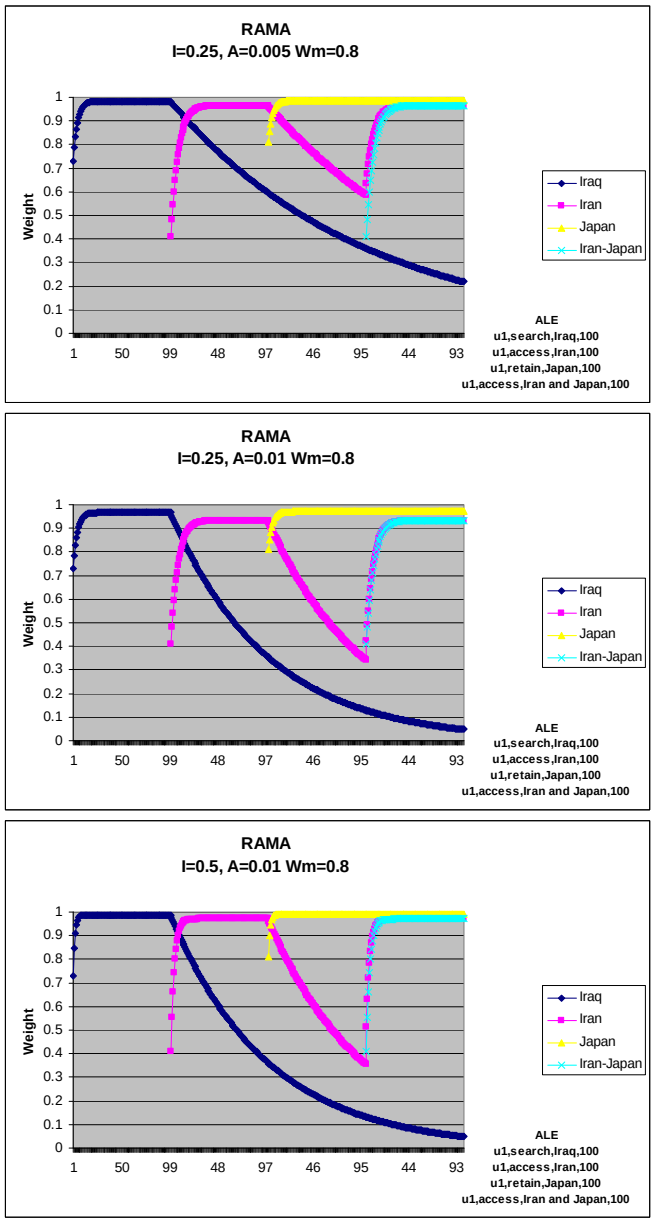

Figure 12. Model adaptation with RAMA using repeated ALEs.

2.1.3.4.2 SIMULATED USER EXPERIMENTS WITH SYNTHETIC DATA

In a simulated experiment, we use a known user model (target model) to generated a fixed number of ALEs and feed them to the model adaptation algorithm. The adapted user model (simulated model) is then compared to the target model to determine their similarity. The more similar the simulated model is to the target model the better the performance of the algorithm. Note in this type of experiment, the simulated model is built for a "simulated user", not a real user.

Given a data set, an ALE can be generated as follows. First draw a small sample from the data set. Then we use the target model to rank the documents in the sample. Next we pick the top-ranked document and make it a selection event made by the "simulated user". Finally we format this selected document as an ALE created by the "simulated user".

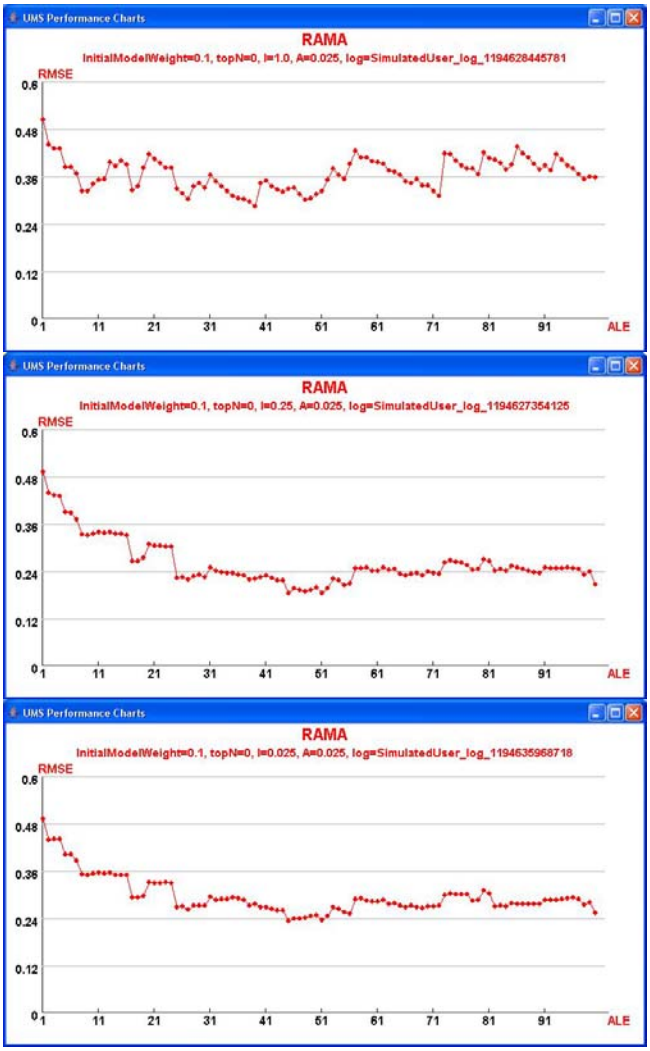

Figure 13. Sample simulated user experiments with A=0.025, top: l=1, middle: l=0.25, bottom: l=0.025.

We use two metrics to measure the model similarity. The first is *the root mean squared error (RMSE)*. This measure aims at the physical resemblance of the models. RMSE is the square root of the mean squared error (MSE). MSE is the average of the square of the "error" across all parameters of the target user model. The error is computed as the weight difference between same concepts/relations in the target and the simulated model.

The other metric is *precision at n (PAN)*. PAN is the precision of the top n documents selected by the current user model from a test data set. The ground truth is the n most relevant documents selected by the target model from the test data set. Thus this measure focuses on the functional similarity of the target and the simulated model.

The simulated model can be initialized empty or with the same structure (concepts and relations) as the target model but different weights. In the case of starting with a simulated model empty, the error for a concept/relation existing in the target model but not simulated model is conservatively assumed to be 1, the maximum difference possible, because the weight ranges between 0 and 1.

The synthetic data set is automatically generated as follows. A *document set* contains N *synthetic documents*: d1, d2, …, dN and is represented as an InfoPackCluster XML. Once generated it is persisted as an XML file for experiments. A Synthetic document consists of a Random number of *weighted elements* and is represented as an InfoPack XML. Note that a synthetic user model can be generated the same way as a synthetic document. A weighted element is an *element* that is randomly drawn from an *element alphabet*. It has an *importance weight*, which is a random number in the range of [0, 1] to indicate importance in document. An element symbolizes a concept or relation. The element alphabet contains a fixed size of elements each of which is an integer number to represent a concept or a relation.

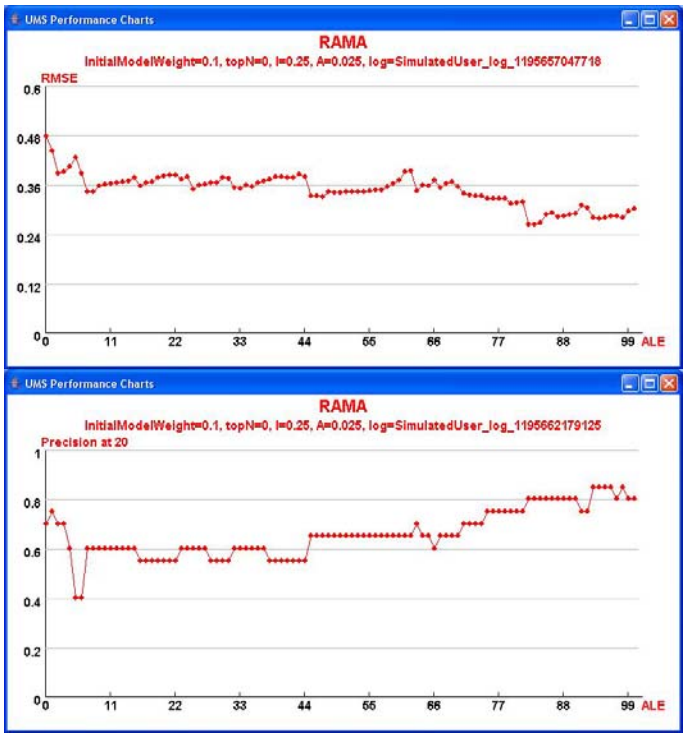

Figure 14. RMSE (top) vs. PAN (bottom) metrics for a simulated user experiment with I=0.025 and A=0.025.

Some sample results are shown in Figure 13.

Here the synthetic data set contains 1000 synthetic documents, each of which has at most 10 elements. The size of the element alphabet is 1000. The target model has 10 elements and random weights. The simulated model is initialized with the same 10 elements as the target model. Their initial weights are all set to 0.1. To generate an ALE, 10 documents are randomly drawn from the data set and then ranked by the target model. The top-ranked document is used to create an ALE. For each experiment, 100 ALE are generated in turn. Each ALE is fed to the RAMA algorithm to evolve the simulated model. The RMSE between the target and simulated model is computed after each round of ALE adaptation. All three panels have the same aging factor value of 0.025 but differ in the reinforcement factor value, which is 1.0, 0.25, and 0.025 for the top, middle, and bottom panels, respectively. We see that the RMSE reaches lowest level in the middle panel.

In another simulated experiment, we compared the two metrics RMSE and PAN (Figure 14). One would expect that the two measures negatively correlate with each other. The two traces indeed show such negative correlation especially second half of the traces.

2.1.3.5 UMS TUNING USING MONTEREY EXPERIMENT DATA

We tuned the RAMA algorithm using the tag|Connect data collected during the Monterey Experiments provided by GDAIS. We describe the preliminary results in this section.

2.1.3.5.1 THE MONTEREY EXPERIMENT DATA

The Monterey Experiments were conducted by GDAIS at Monterey Institute from October 1, 2007 to November 14. It featured a simulated national intelligence analytical setting with data collection, organization and analysis phases. Over 10 students at the Institute participated. The participant was given the following tasking:

"What are the most important and most likely threats to the security of the United States and its interests abroad that could arise from the Islamic Republic of Iran within the next 10 years?"

The participant was expected to deliver a National Intelligence Estimate (NIE). The tools available to the participants include tag|Connect and Catalyst. The ALEs generated by both tools form the data that we plan to use for the purpose of evaluating our user modeling component.

2.1.3.5.1.1 TAG|CONNECT ALE STATISTICS

Total ALEs: 770

Start time: 2007-10-03T18:37:56.767-07:00

End time: 2007-11-14T20:24:12.1018256-05:00

Users: 10 with usable number of ALEs, 5 with too few ALEs. The 10 usable users are: Alethia.Jimenez, Alexander.Billings, Brian.Lushko, Cameron.Stanuch, David.Lettis, Emily.Baker, Jade, Lindsay.Gardner, Paul.Markovs, and Rocco.Costa.

Type of ALEs: Access, Annotate, Retain, SessionEvent, StartApplication

Originating Application URI: urn:gdais:tagConnect

2.1.3.5.1.2 CATALYST ALE STATISTICS

Total ALEs: 2002

Users: 10 (same set as tag|connect) with usable number of ALEs, 8 with too few ALEs.

Type of ALEs: Annotate (10, all created by Lindsay.Gardner), Create, Modify, SessionEvent, StartApplication

Originating Application URI: urn:gdais:Catalyst:test

2.1.3.5.2 TUNING APPROACH

2.1.3.5.2.1 METRIC

We use RMSE (root mean square error) and/or PAN (precision at N). For PAN, we will preserve some ALEs to build the test data with ground truth. That is, we will pool the relevant documents from ALL users. The relevant documents from a particular user will be the ground truth for that user.

2.1.3.5.2.2 USER MODEL BUILDING

Whole-data model: We build a model with all available ALEs from a domain for a user.

Half-data model: We partition the ALEs from a domain for a particular user into two halves with one half occurring before the other based on the publication time. We then build a model with each half.

Model adaptation algorithm RAMA or BUMA are used.

2.1.3.5.2.3 MODEL EVALUATION

2.1.3.5.2.3.1 ACROSS APPLICATION DOMAIN

Compare the Whole-data models built using tag|Connect events with those using the Catalyst events. We predict that the similarity between the two domain models for the same user will be greater than that for two different users.

User	tag|Connect	Catalyst	RMSE
Alethia.Jimenez	Maj-t	Maj-c	Raj-aj = RMSE(Maj-t, Maj-c)
Alexander.Billings	Mab-t	Mab-c	Rab-ab = RMSE(Mab-t, Mab-c)
…8 more users			

Compute the following correlation matrix:

	Alethia.Jimenez **Maj-c**	**Alexander.Billings** **Mab-c**	**Brian.Lushko** **Mbl-c**	**…8 more users**
Alethia.Jimenez **Maj-t**	Raj-aj	Raj-ab	Raj-bl	…
Alexander.Billings **Mab-t**	Rab-aj	Rab-ab	Rab-bl	…
Brian.Lushko **Mbl-t**	Rbl-aj	Rbl-ab	Rbl-bl	…
…8 more users	…	…	…	…

We predict that the RMSE on the diagonal will be larger than the non-diagonal cell values that are on the same row or column.

2.1.3.5.2.3.2 ACROSS TIME DOMAIN

Compare half-data models for the same application domain. We predict that the similarity between the two domain models for the same user will be greater than that for two different users.

User	First Half	Second Half	RMSE
Alethia.Jimenez	Maj-1	Maj-2	Raj-aj = RMSE(Maj-1, Maj-2)
Alexander.Billings	Mab-1	Mab-2	Rab-ab = RMSE(Mab-1, Mab-2)
…8 more users			

Compute the following correlation matrix:

	Alethia.Jimenez Maj-2	**Alexander.Billings Mab-2**	**Brian.Lushko Mbl-2**	**…8 more users**
Alethia.Jimenez Maj-1	Raj-aj	Raj-ab	Raj-bl	…
Alexander.Billings Mab-1	Rab-aj	Rab-ab	Rab-bl	…
Brian.Lushko Mbl-1	Rbl-aj	Rbl-ab	Rbl-bl	…
…8 more users	…	…	…	…

We predict that the RMSE on the diagonal will be larger than the non-diagonal cell values that are on the same row or column.

2.1.3.5.3 TUNING EXPERIMENTAL DESIGN

For this tuning activity, we use the tag|Connect data from the Monterey Experiments only. This data contains a total of 2002 ALEs generated by 10 users (i.e. analysts). The types of ALE we processed are Access, Retain, and Annotate.

The modeling adaptation algorithm being tuned is RAMA. The settings of the algorithm are as follows. The reinforcement factor and the aging factor values are 0.25 and 0.025, respectively. The maximal number of new concepts allowed to insert for given ALE is 5.

For each user, we split her ALEs in 2 halves in order of their timestamp. We use RAMA to build a model for each half. With 10 users, we create 20 half data models. For each half data model, we compare its similarity with every one of the 20 half data models. We then identify the 5 half data models (other than itself) that are most similar to the given model. The similarity between two models is defined as (1-RMSE), where RMSE is the root mean square error between the two. The hypothesis is that the 2 halves of the same user should be more similar to each other than to other models.

In addition, we split each user's ALEs in quarters in order of their timestamp. We then use RAMA to process them similar to the half data. The hypothesis is that the quarter data models of the same user should be more similar to each other than to other quarter data models.

2.1.3.5.4 HALF DATA MODEL RESULTS

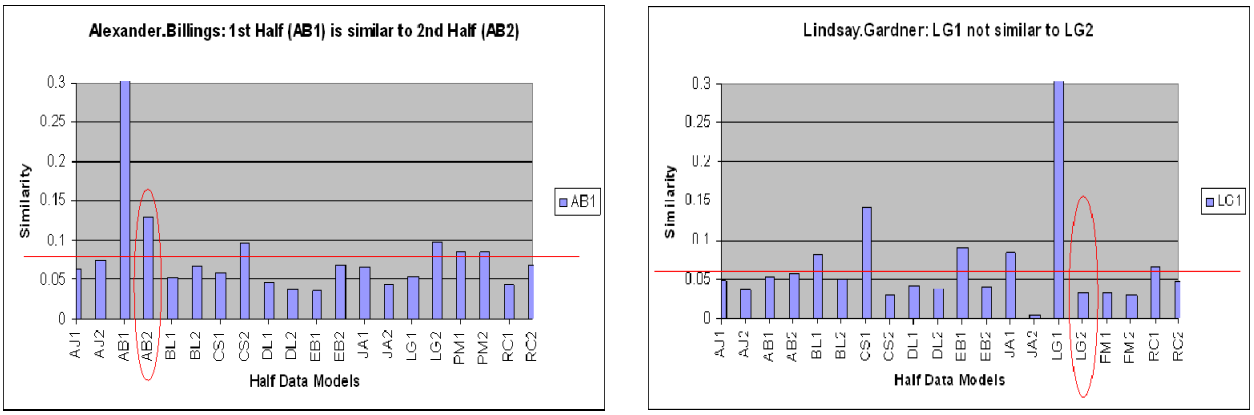

Figure 15. Left: user Alexander's half data models (AB1 and AB2) are similar to each other. Right: user Lindsay's half data models (LG2 and LG2) are not similar to each other.

In 4 out of the 10 users, their half data models are among the top 5 most similar models. One example is given in left panel of Figure 15. The top 5 most similar models for user Alexander's first half model (AB1) are AB2, CS2, LG2, PM1, and PM2 as indicated by being above the red horizontal line in the figure.

For the other 6 users, their half data models are not among the top 5 most similar models. One example is given in right panel of Figure 15. The top 5 for user Lindsay's first half model LG1 does not include LG2.

2.1.3.5.5 QUARTER DATA MODEL RESULTS

For all 10 users, at least one pair of their quarter data models are among the top 5 most similar models. A good example is user Brian's models (top panel in Figure 16) where the first quarter is similar to all other 3 quarters. The bottom panel of Figure 16 shows the worst instance in the pack. Even here, the user Rocco's first quarter is similar to the fourth quarter model (i.e. the 4th quarter model is one of the top 5 for 1st quarter model).

Thus it seems that with finer data splits, the split data models of the same user show better similarity. In other words, the quarter data results seem to confirm our original hypothesis better than our half data models. We think there are two reasons for the observation. First, all 10 users are working on the same tasking. It is understandable that different users show similarity. Second, the time span for the original experiment is 6 weeks, during which the user may shift their current interest from time to time. The farther apart in time, the more shifts in the interests.

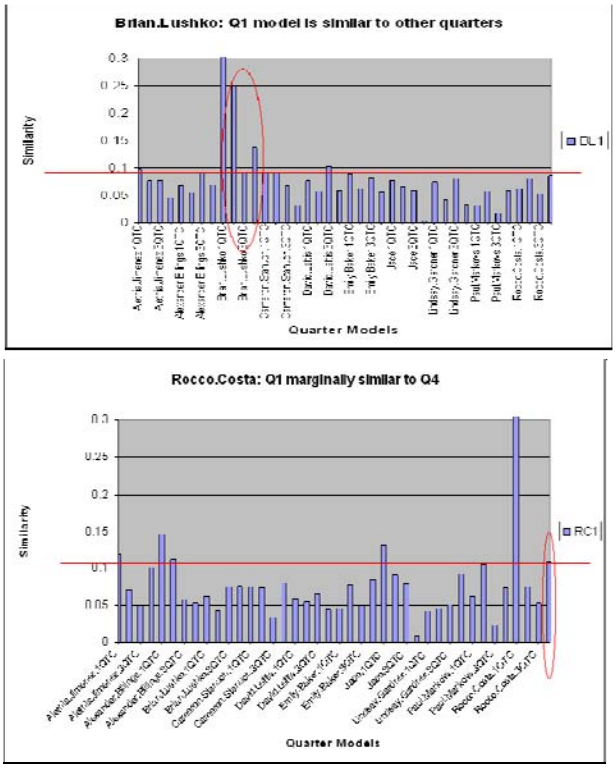

Figure 16. Top: user Brian's quarter data models are similar to each other. Bottom: user Rocco's quarter models are marginally similar to each other.

2.1.3.6 EXTENDING UMS TO DIFFERENT OBJECT MODELERS

The UMS algorithms were initially developed using T2SCM and TF object modelers. In particular, the two modelers were used in the first step of both BUMA and RAMA algorithms to extract concepts and/or relations from the text content of the feedback. For the NIST final evaluation, we have the opportunity to work with other types of object modelers: LDA, PAM, and TAE (see Section 2.1.2.3). The extension was fairly straightforward. We simply dropped the new modelers in place of the T2SCM or TF in the first step of the RAMA algorithm (note RAMA is the only user modeling algorithm used in the final evaluation). The other steps of the algorithm remained the same as before. The markers extracted by the new modelers were converted to concepts using the following mapping:

- Marker ID -> concept ID
- Marker Label -> concept name and definition
- Marker Weight -> concept weight

2.1.3.7 AUTOMATIC QUERY GENERATION FROM USER MODEL

For the NIST evaluation, it was necessary to develop an approach to automatically generate a query from a user model. In other words, the query should be relatively short and yet truly representing the user model. In the evaluation, the query was used to retrieve Web documents for the DR evaluation. We have discovered that the query generation from a user model is a complex issue and warrants scientific research in its own right. We will discuss this issue a bit more in the future research section. To make do for the evaluation, we have come up with a simplistic “engineering hack” described below.

Our approach was to take the first term of the N markers (i.e. concepts) with highest marker weight and then make an implicit “AND” query by placing a space between these terms. We have also made sure that there were no duplicate terms in the query and no terms with non-ASCII characters.

For example, the 6 top-weighted markers with 3 terms each in user saniya33's model are listed below in descending order of weight:
1. chemical biological weapons
2. united world international
3. russia russian soviet
4. missile nuclear missiles
5. nuclear north iaea
6. smallpox anthrax biological

The query automatically produced by the query generation algorithm is: *[chemical united russia missile nuclear smallpox].*

2.1.3.8 UMS SOFTWARE

The purpose of the UMS software is to build, store, and provide models of users, as well as build and provide Virtual Interest Groups (VIGs).

The UMS passively consumes InfoPacks, which are ALEs that have been fully processed by the ALSClient as well as online object modelers and provided via XMLBLASTER. These InfoPacks used for user model adaptation according to object model type (i.e. TF, T2SCM, LDA, PAM, TAE).

To obtain current user models, other InformANTS components query the UMS, which provides the models via XMLBLASTER. Likewise, the UMS can provide user model queries, which are strings of a request weight derived from the most prominent markers in a user model. Upon request, the UMS can also provide a VIG for a user, which is a ranked list of the users most similar to a user.

In addition to XMLBLASTER, the UMS can also support webservice calls for user models and VIGS, although this service was not a part of the final version of the system.

2.1.4 INFORMATION MATCHING SYSTEM (IMS)

Following the hyper-spectral modeling approach (section 2.1.1.1), the Information Matching System (IMS) in InformANTS analyses the various object models that describe the content of a document or the current interest of a user and arranges the InfoPacks of these documents and users in a common information space where relative distance represents relevance. So, the closer two InfoPacks are in this space, the more relevant they are to each other. From this arrangement, the IMS then derives the InformANTS products, such as recommendations of documents for a user or reordering of documents recommended by Google based on a user query.

The main components of the IMS are:

- A collection of Object Model Reasoners (short “reasoners”) that assess the similarity of object models of the same kind stored in different InfoPacks,
- An InfoPack Locator (short “locator”) that arranges the InfoPacks in its information space based on input from the object model reasoners,
- An InfoPack Database that maintains an up-to-date representation of all InfoPacks for tracking and logging purposes.

The main flow of information in the IMS is shown in protocol 1 (Figure 4) and protocol 3 (Figure 5). Any announcement of new or updated InfoPacks is processed by the reasoners, which select the object model from the InfoPack that matches their respective model type (modeling approach chosen for the object model, section 2.1.2.3). If the InfoPack carries an object of this type, the reasoner computes the similarity of this model with all other previously announced InfoPacks (with that model type) and in turn announces a "force" to the locator that defines the similarity-attraction between these two InfoPacks. The locator integrates the forces among all InfoPacks in an iterative force-based arrangement process, which results in self-organizing patterns of InfoPacks in information space. These patterns adjust as new InfoPacks "arrive" in the IMS (e.g., new page viewed by user, new user logged on) or as the models in existing InfoPacks change (e.g., user interest model updated, (wiki) page edit saved).

In the following, we discuss the operation of the main components of the IMS.

2.1.4.1 IMS COMPONENT: OBJECT MODEL REASONERS

In a particular run of the InformANTS system, the software is configured to instantiate a specified set of object modelers in the IOMS (section 2.1.2.3). Symmetrically, the IMS will instantiate corresponding object model reasoner processes either in the same Java Virtual Machine or (through an appropriately configured startup-script) on a different host (connected through the xmlBlaster infrastructure). The reasoner for a particular model type implements a similarity measure for model instances of this type. For instance, the SCM reasoner (section 2.1.4.1.1) compares models created by the T2SCM object modeler (section 2.1.2.3.1). From these similarities, the reasoner derives the strength of an attractive force that is applied in the locator. The more similar two object models carried by two InfoPacks are, the stronger the attractive force is acting between the two InfoPacks in the locator.

In the following, we discuss the similarity-to-force calculation for each modeling type that was implemented for the final NIST evaluation.

2.1.4.1.1 IMS COMPONENT: OBJECT MODEL REASONERS – SCM

The Text-to-Specialized-Concept-Maps (T2SCM) object modeler in the IOMS creates maps of concepts of specific types through entity extraction where each entity carries a number of characteristics (attributes) that provide known background information about the entity. The CASE prototype is restricted to "Place" and "Person" concepts, where each place may carry latitude and longitude characteristics, and each person may be described by the date of birth and the gender. The relations between these concepts in the Specialized-Concept-Maps (SCM) are limited to co-occurrence in the corresponding text. Many more characteristics and concept types may be possible if sufficient data for the characteristics and enhanced entity extraction methods were available.

The Specialized-Concept-Map (SCM) reasoner communicates forces between pairs of InfoPacks to the locator where the strength of the force between two InfoPacks is proportional to the similarity of their SCM models (if they carry any). In the following, we first discuss a sophisticated method for determining these forces that we deployed in early InformANTS prototypes. Then we discuss a more simplistic approach that we chose in the NIST evaluation when it became clear that the entity data available to our team lacks the richness to justify the computational effort of the sophisticated method.

2.1.4.1.1.1 COMPLEX SCM REASONER

The complex SCM reasoning mechanism deployed for instance in the CASE IE3 demonstration, operates in three steps that are repeated within each IMS cycle:

1) Train Self-Organizing Maps (SOM) against the current set of SCM concepts and assign each concept a Concept Proximity Space location by mapping through the SOM

2) Adjust the concepts' proximity space locations according to their SCM relationships

3) Estimate the similarity of SCM's carried by different InfoPacks from the adjusted concept locations and calculate an attractive InfoPack force for the IMS locator component

2.1.4.1.1.1.1 SELF-ORGANIZING MAPS (SOM) FOR SCM DIMENSIONALITY REDUCTION

In the first step of the SCM similarity-to-forces calculation, we reason over the characteristics of the concepts that are held by the InfoPack population, training and using a separate Self-Organizing Map (SOM) for each specialization type of a concept (i.e., Place, Person, Event, or Object). The SOM is a rectangular array of nodes that are embedded in a one-by-one box in Euclidian (2D) space. Each node of the SOM carries a "Model", which is a data structure that holds variables for those characteristics of the concept type that are used in the mapping of the concepts into their Proximity Space. For instance, the Models of the nodes of the SOM that maps Place concepts carry Place characteristics, such as longitude, latitude, or altitude.

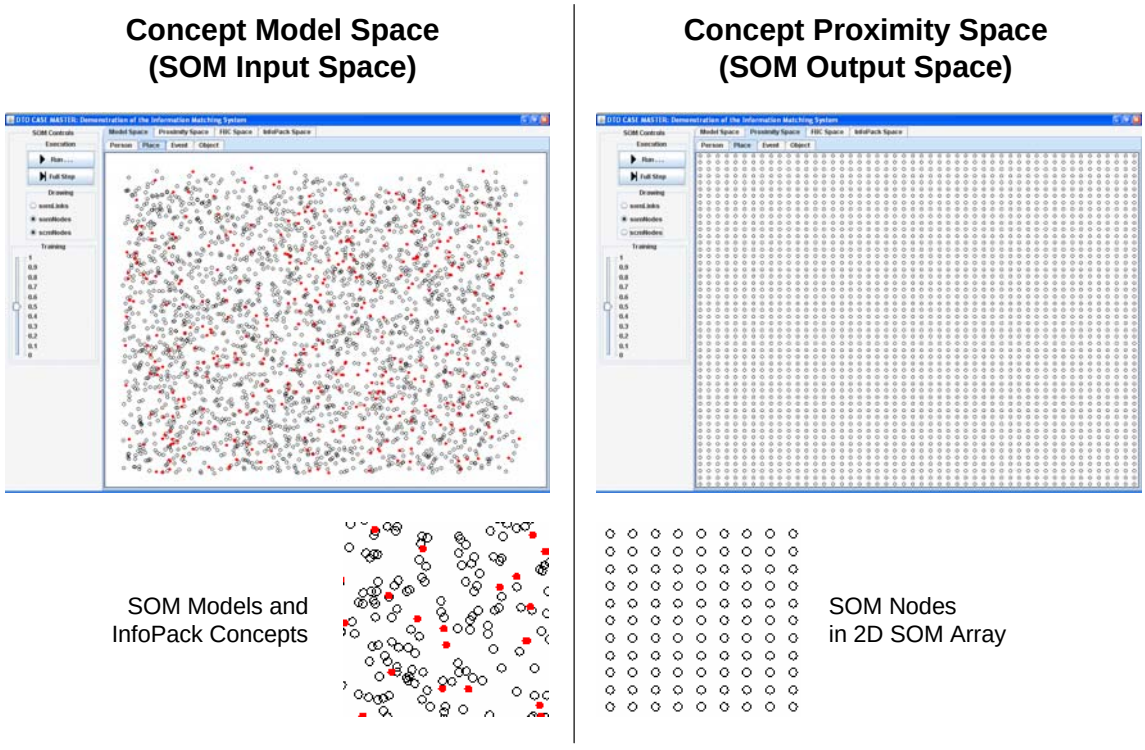

Figure 17. SOM's (black circles) map from the space of concepts (red circles) to a 2D Euclidian space.

Initially, the Models of the nodes in the SOM are assigned random values for their characteristics (Figure 17), but the emergent effect of the training process of the SOM is that the SOM nodes (black circles) cluster close to the concepts from the InfoPacks (red circles) and that the neighborhood relationships of the SOM array are approximated by the distance neighborhoods of the Models carried by the nodes. Figure 18 shows the result of the training process.

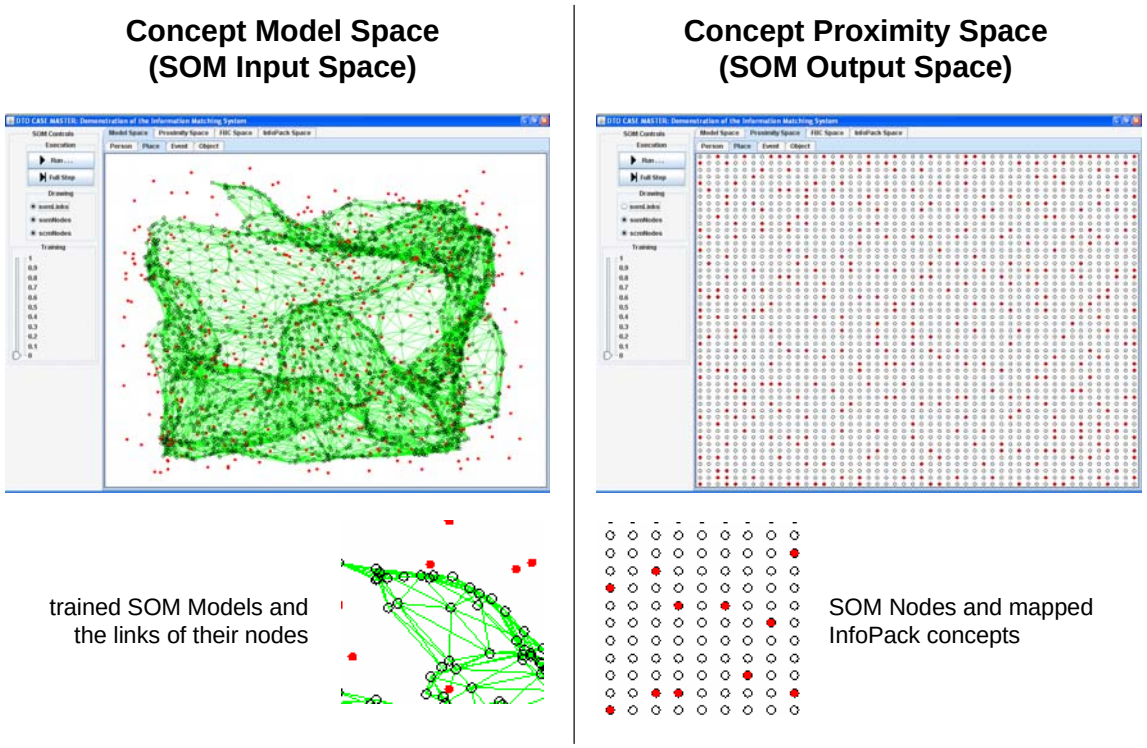

Figure 18. A trained SOM embeds the array neighborhood structure in the model space.

Any node in a SOM has a static location in the one-by-one box in Euclidian Proximity Space. We map a concept from its model space spanned by the characteristics associated with this concept type into the Proximity Space in two steps. First, we search in the SOM for the node that carries the most closely matching model. This node is called the "Best Matching Unit (BMU)". Then, we assign the concept the same Proximity Space coordinates as its BMU.

2.1.4.1.1.1.2 FORCE-BASED CLUSTERING (FBC) FOR SCM RELATIONS

The location of a concept as suggested by the SOM is purely determined by the concept's characteristics and the characteristics of all the other concepts of the same type in the InfoPacks in the IMS. In particular, it is independent of any relationships that may link some of the concepts in an InfoPack together. Therefore, we use Force-Based Clustering (FBC) to integrate this additional information about a document or analyst into the Information Exploration process.

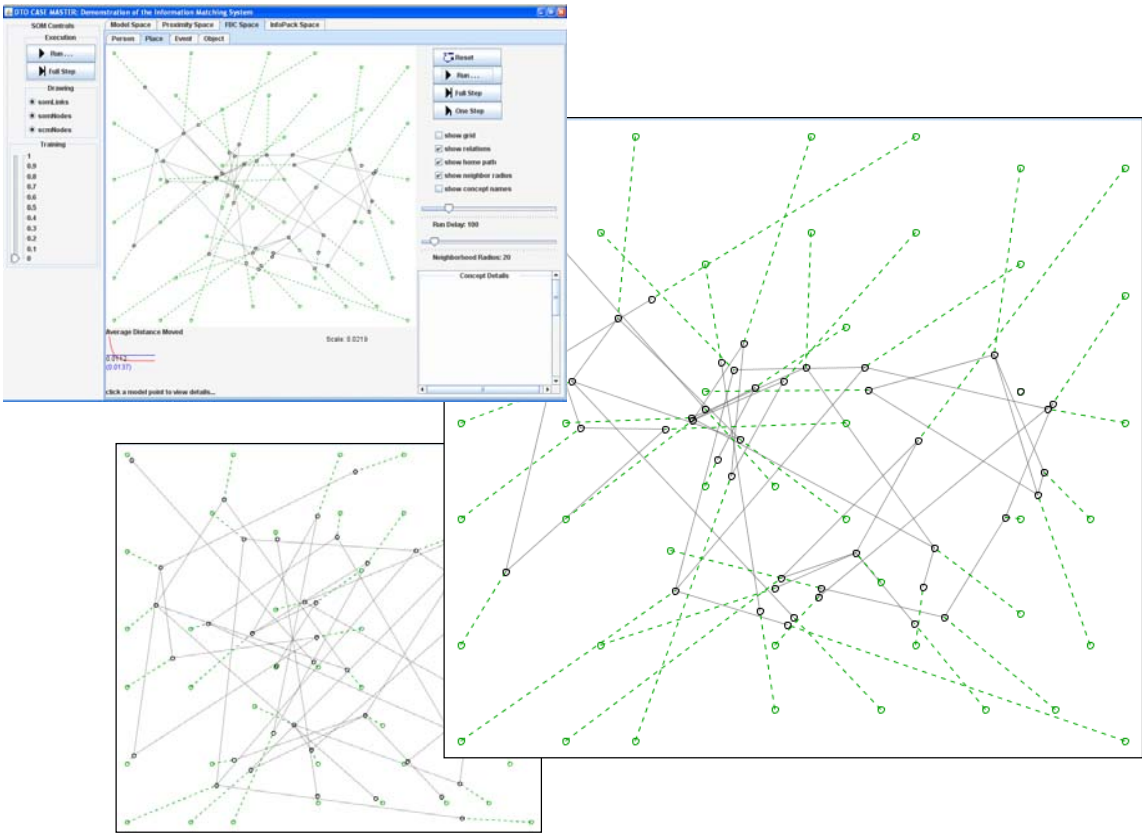

Figure 19. Force-Based Clustering reasons over relations between concepts embedded in Proximity Space.

In the FBC algorithm, each concept has two locations in Proximity Space. The “home” location is the one currently suggested by the SOM, which may change as the SOM evolves. The “current” location is the location updated by the FBC. The FBC process continuously iterates over all concepts in the IMS, calculating updates to their current location. In the location update, the concept balances its attraction (back) to its “home” location with its attraction to the “current” location of other concepts that it is linked to in relations in the InfoPack. The quantitative strength of these attractions is determined by the weights associated with the relations. If the “home” location suggested by the SOM and the relation links stored in the InfoPacks do not change, then the FBC algorithm will converge on an “energy-minimizing” configuration in which all the attractions are in balance. If the input conditions change, then the FBC will approach a new balance, thereby responding dynamically to the changing knowledge held in the IMS.

2.1.4.1.1.1.3 PROXIMITY SPACE DISTANCE TO LOCATOR FORCE CALCULATION

The SOM and FBC mechanisms determine a (dynamically updated) x/y location in the 1-unit-by-1-unit Concept Proximity Space for each concept in the SCM's held by any of the InfoPacks currently known to the system. We "roll up" the concept-neighborhood relations in the SCM reasoner in this space into a similarity force that is communicated to the InfoPack locator.

For each InfoPack that carries an SCM object model, we first determine the weighted center of gravity for all concepts of the same type (Place or Person). The weights in this calculation are the importance levels that are set for each concept by the T2SCM (document InfoPack) or the UMS (user model InfoPack). Then we determine for each InfoPack the set of all other InfoPacks that are within a certain radius (experimentally determined) from the center of gravity within the Concept Proximity Space of the given type. For each of these neighboring InfoPacks, we calculate a type-specific attractive force whose strength is inversely proportional to the distance to the InfoPack (farther away InfoPacks exert less attraction). Finally, we add all force vectors for each type of Concept Proximity Space into an aggregated attractive force between pairs of InfoPacks and communicate this force to the InfoPack locator component of the IMS.

2.1.4.1.1.2 SIMPLE SCM REASONER

In the simple SCM reasoner deployed in the final NIST evaluation, we calculate forces between InfoPacks directly from the characteristics carried by the SCM object model of the InfoPacks in the IMS. For each InfoPack, we iterate over all concepts in its SCM (if it has any), and determine attractive component forces to concepts of other InfoPacks. These component forces are derived from the characteristics carried by the concept (lat/lon for Place, birth-date/gender for Person). Concepts (from other InfoPacks) that have similar characteristics are assigned stronger forces.

For each InfoPack, we compute the vector sum of all component forces to concepts of another InfoPack and communicate the resulting inter-InfoPack forces to the IMS locator component.

2.1.4.1.2 IMS COMPONENT: OBJECT MODEL REASONERS – TF

The Term-Frequency (TF) reasoner communicates forces between pairs of InfoPacks to the locator where the strength of the force between two InfoPacks is proportional to the similarity of their TF models (if they carry any).

The InfoPacks carry the document object models as generated by the object modelers described above. The method to calculate the force between two document models generated by a topic modeler is the same for all four types of (strict and notional) topic modelers (TF, TAE, LDA, and PAM). As described above, each document model is represented by a single point in the abstract document space of each model type. The proximity of any two points in this space is directly proportional to the similarity between the documents which are represented by those points. To quantify that similarity, we subtract from one the normalized distance between the points. Because the models are considered to be probability distributions, we chose a distance measure appropriate for probability spaces. The Hellinger Divergence is given by

$$d_H = \sqrt{\frac{1}{2}\sum_i (\sqrt{p_i} - \sqrt{q_i})^2} \,,$$

where the sum is over all markers in the marker set. p_i and q_i are the marker weights for the two different documents. Because the marker weights are normalized, this quantity will necessarily be in [0,1]. The strength of the force between the two documents, as given by

$$force = 1 - d_H \,,$$

will necessarily be in [0,1] as well.

2.1.4.1.3 IMS COMPONENT: OBJECT MODEL REASONERS – TAE

The Text Analysis Engine (TAE) reasoner communicates forces between pairs of InfoPacks to the locator where the strength of the force between two InfoPacks is proportional to the similarity of their TAE models (if they carry any).

2.1.4.1.4 IMS COMPONENT: OBJECT MODEL REASONERS – LDA

The Latent Dirichlet Allocation (LDA) reasoner communicates forces between pairs of InfoPacks to the locator where the strength of the force between two InfoPacks is proportional to the similarity of their LDA models (if they carry any).

2.1.4.1.5 IMS COMPONENT: OBJECT MODEL REASONERS – PAM

The Pachinco Allocation Model (PAM) reasoner communicates forces between pairs of InfoPacks to the locator where the strength of the force between two InfoPacks is proportional to the similarity of their PAM models (if they carry any).

2.1.4.2 IMS COMPONENT: INFOPACK LOCATOR

Our Information Matching System (IMS) continuously (re-)organizes a dynamically changing set of user and document models in an abstract information space such that models that are similar to each other remain close in that space while those models that are not similar tend to be farther away. The InfoPack Locator is the component that hosts the self-organizing process while the object model reasoners (section 2.1.4.1) provide the information (similarity forces) that drives this process.

In the following, we review the atomic element of the self-organizing process (the InfoPack), discuss the topology of the information space in which the InfoPacks are organizing, and present how the self-organizing process operates.

2.1.4.2.1 WHAT IS SELF-ORGANIZED?

For the purpose of the matching process we unify user and document models (both comprise a set of object models from the IOMS) under the term "InfoPack". Architecturally (Figure 20), an InfoPack consists of a set of meta-level attributes that describe, for instance, its origin (a particular user ID or the URL of a document) or the level of attention that the system should give it. The InfoPack also carries a list of object model instances, each of which carries model data and an identifier of the specific object modeler type that generated it.

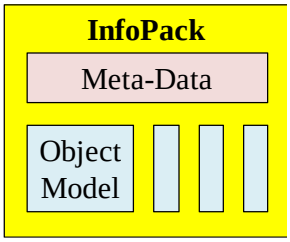

Figure 20. User and Document Models are InfoPacks.

Any given InfoPack relates to a specific entity outside of InformANTS. Each user model InfoPack corresponds to a particular analyst that has been or is currently working in the front-end system (nSpace2 in NIST evaluation). Each document model InfoPack contains the object models for a particular document accessible in the corpus. When the contents of a document changes (as it is for instance the case with Wiki pages) or as the user interest evolves, the object models in the corresponding InfoPacks are updated. If new documents or users "arrive" new InfoPacks are created for them.

2.1.4.2.2 WHERE DOES SELF-ORGANIZATION HAPPEN?

Each InfoPack has a unique location in the IMS' information space that may change over time. The information space is the surface of a torus (Figure 21) that we "create" by either wrapping the edges of a 1x1 square (2D coordinates) or the sides of a 1x1x1 box (3D coordinates). Our initial prototypes used a 2D torus, but recently we decided to move to 3D to provide the InfoPacks with a larger degree of freedom in their movement. The torus surface is necessary to avoid edge effects that would otherwise distort the relative distances between InfoPacks.

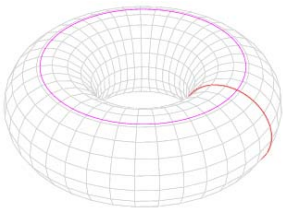

Figure 21. The Information Space is a Torus.

The absolute distance between InfoPacks is defined as the distance on the torus. Since our 2D or 3D torus is created by wrapping a square or a box, we can easily calculate the surface distance from the minimum of all distances between one InfoPack's location (Blue in Figure 22) and the enumeration of all wrapped locations of the other InfoPack (Red in Figure 22).

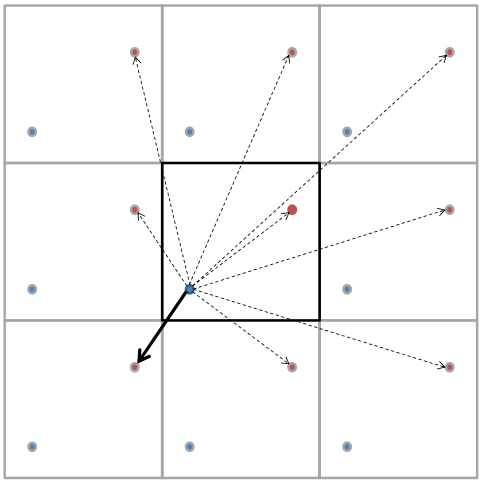

Figure 22. 2D Torus Surface Distance Calculation.

2.1.4.2.3 HOW DOES SELF-ORGANIZATION HAPPEN?

In the locator, the InfoPacks are agents that move in information space (torus), driven by local and non-local interactions with other InfoPacks.

The desired emerging system-level behavior of this agent system is stated as follows:

- Arrange InfoPacks on the torus according to their perceived overall similarity.
- Similarity of two InfoPacks is defined by the object models that they carry. If two object models of the same type are similar in an appropriate similarity metric, then this measure should contribute to the overall similarity of the InfoPacks.

From this global goal, we derive the following agent "intentions":

1) InfoPacks that are not known to be similar do not "want to" be close in information space.
2) InfoPacks that are known to be similar "want to" be close in information space.

Then we translate these intentions into notional forces that drive each InfoPack's movement from its current location in information space:

1) Repulsive Force (local): An InfoPack in information space is pushed away from neighboring InfoPacks with a strength decreasing by distance.
2) Attractive Force (non-local): An InfoPack in information space is pulled towards any InfoPacks that are known to be similar.

On the one hand, the repulsive force on an InfoPack's location is based directly on the distances on the torus (local interaction). The closer two InfoPacks are on the torus, the stronger (longer force component vector directed away from the repelling InfoPack) should this force be. For computational simplicity, we cut off this force at a certain distance on the torus. Without the presence of any attracting force components the indiscriminate repulsive forces result in an arbitrary homogeneous arrangement of all InfoPacks across the torus.

On the other hand, the strength of the attractive forces on an InfoPack's location is not determined by the distance relationships on the torus (non-local interaction). Rather, their strength is modulated by the similarity of InfoPacks as perceived from the perspective of a particular object model type carried by the InfoPacks. Thus, InfoPacks may be attracted to any other InfoPack on the torus regardless of their location, as long as these two InfoPacks carry similar object models.

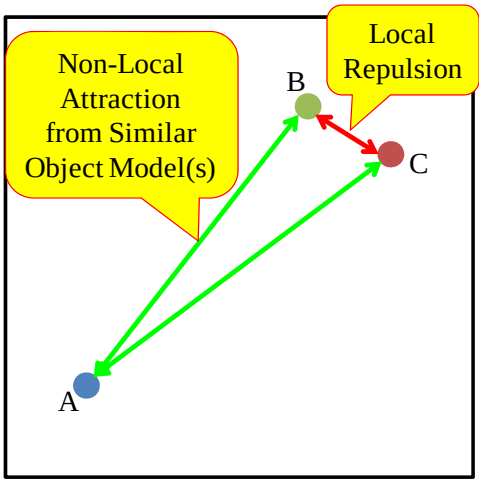

Figure 23. Three InfoPacks driven by Attractive and Repulsive Forces.

Any InfoPack will apply attractive force components (towards other InfoPacks) if these InfoPacks are similar in any of their object models. If the InfoPacks are similar across multiple object model types, then more than one attractive force component is calculated. The strength of the force (length of the component vector towards the other InfoPack's location in the torus) is proportional to their similarity for a particular object model type. Thus the more similar the object models are, the stronger the attractive force will be. Figure 23 illustrates these forces in a 3-InfoPack example.

An InfoPack will calculate all attractive and repulsive force components as a function of its current location on the torus and the object-model similarities of all other InfoPacks. Once the weighted sum (weights controlling the importance of the two counteracting intentions) of all these attractive and repulsive force components is computed, the InfoPack will move a length-limited step into the direction of this combined force.

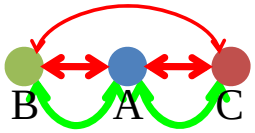

Figure 24. Converged State.

This movement process is repeated for all InfoPacks in the locator indefinitely, but it approaches relative convergence as the pattern of attractive and repulsive forces balances out in a minimum energy state. Figure 24 shows a possible energy-minimizing state for the InfoPacks in Figure 23.

2.1.4.2.4 FORCE-BASED ARRANGEMENT OF INFOPACKS

The various object model reasoners in InformANTS communicate forces based on pair-wise InfoPack similarity for a particular object model type to the locator. A force is described by four attributes: the object model type, the id's of the two InfoPacks, and the strength of the force (between -1 and +1). The locator updates the list of known forces for all InfoPacks in the system.

The locator is the execution environment for the InfoPack agents that move their InfoPacks on the surface of the torus that makes up the information space. It creates new agents when new InfoPacks arrive or removes them when the InfoPack is deleted, and it continuously iterates over the list of agents to give each a chance to update its location based on the current force pattern.

An InfoPack agent, when activated by the locator, iterates over the list of forces that include its InfoPack's id. For each InfoPack that is also part of such a force, it determines the shortest vector on the torus to the current location of the InfoPack. This vector provides the direction of the force, but the strength of the force and whether it is attractive (positive strength) or repulsive (negative strength) is set by the reasoner that communicated this force. The agent scales the vector accordingly and averages all these force-component vectors. *The resulting vector represents the combined guidance by all reasoners.*

In a second step, the agent iterates over all InfoPacks in its immediate neighborhood (up to a given threshold) on the torus. For all these neighbors, it averages repulsive component force vectors that point away from the InfoPacks and whose length is inversely proportional to their distance to the agent's InfoPack. *The resulting vector represents the general uncertainty or lack of knowledge of the relationships between InfoPacks.*

Finally, the agent adds the guidance vector from the reasoners and the uncertainty vector from its neighborhood to determine the direction of its next step on the torus. If the length of this vector is below a globally defined step-length threshold, then the next step will be limited in length to allow the InfoPacks to settle down in equilibrium. Otherwise, the agent will take a step equal to this threshold in the direction of this vector. Limiting the length of the agent's move allows the system as a whole to gradually evolve towards equilibrium on complex individual trajectories, avoiding thrashing and other pathological emergent patterns.

2.1.4.2.5 EXPERIMENTS

Designing self-organizing processes with emergent functionalities, such as the force-based arrangement of InfoPacks in the locator, is still more of an art than a science. It takes experience and intuition to define the right set of local behavioral rules (e.g., responses to forces) to achieve the desired global behavior (see section 2.1.4.2.3). Therefore, it is prudent to first try out the mechanism in an abstract and simplified implementation before taking it to production.

In the following, we first present two intuition-confirming experiments before we then discuss how we confirmed the usefulness of the self-organizing process in the setting of the NIST evaluation.

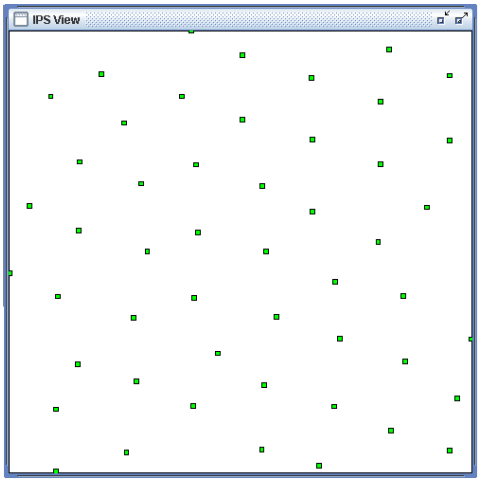

Figure 25. Homogeneous Dispersion.

2.1.4.2.5.1 ABSENCE OF GUIDANCE EXPERIMENT

We claim in section 2.1.4.2.3 that absent any guidance forces from the reasoners, the localized repulsive uncertainty forces will result in an arbitrary but approximately homogeneous distribution of InfoPacks across the torus. Experiments with 40 InfoPacks without any active reasoners indeed show no distinguished clusters (Figure 25).

2.1.4.2.5.2 LARGE ARTIFICIAL DATASET EXPERIMENT

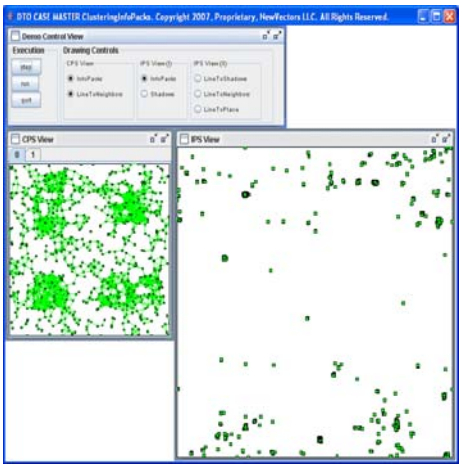

Figure 26. Four Clusters in the Artificial Dataset - Ground Truth.

In another experiment we created 1000 artificial InfoPacks with only an SCM object model where each InfoPack's model carried only one person and one place entity. The models for each InfoPack were created such that the dataset contained four clusters (with some spread) of 200 InfoPacks each. The remaining 200 InfoPacks were created randomly across the entire data range. Figure 26 shows the "ground truth" in the data, where the black dots are the single place entities in the SCM models of the InfoPacks and the green lines are connecting entities that are considered close enough by the SCM object model reasoner to communicate attractive forces to the locator.

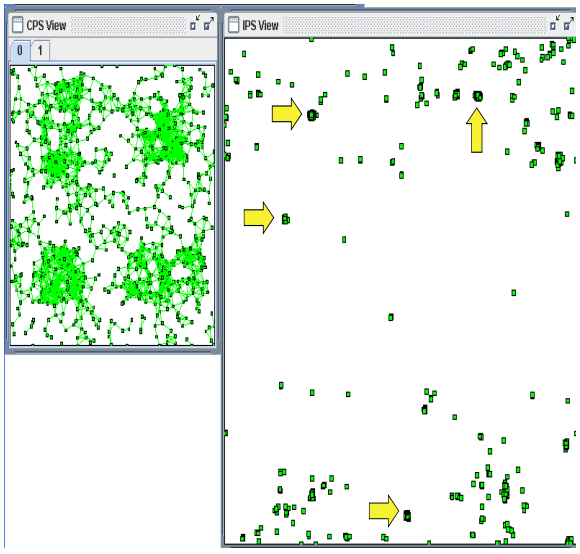

Figure 27. Clustering of 1000 Artificial InfoPacks.

We ran the experiment with these 1000 InfoPacks for approximately 30 seconds on a standard PC. Figure 27 shows the results with the main clusters highlighted. The artificial noise in the data created more than four clusters, but the majority of the InfoPacks do fall closely together as expected.

2.1.4.2.5.3 RECOMMENDATION OVERLAP EXPERIMENTS

The final purpose of the self-organizing process in the locator is to arrange the InfoPacks such that similar InfoPacks are closer to each other than less similar ones. Of course, divergent similarity "opinions" for the same InfoPack pairs could be communicated from competing reasoners (if there is more than one), or the similarity associations among the InfoPacks could form a graph that cannot be embedded in a low-dimensional space such as our 3D torus (Figure 21). But for the Information Matching System to provide any useful recommendations to the users, there needs to be at least some overlap between the "raw" pairwise similarities of the object models in the InfoPacks and the emergent arrangement of the InfoPacks in the locator.

In the following, we discuss a metric that measures this overlap pragmatically from the perspective of the re-rank requests to which the IMS responds and we present some performance measures with this metric.

2.1.4.2.5.3.1 RE-RANK RECOMMENDATION OVERLAP

In the NIST evaluation experiment, the most important service provided by the InformANTS system was the re-ranking of Google-recommended documents by relevance (and novelty) relative to a given user model. To answer this request, InformANTS would download, model, and create InfoPacks for any of the Google-recommended documents that were not yet in the system. The continuous self-organizing process would arrange the user and document InfoPacks on the 3D torus, and when the response timeout for the re-rank request is reached, InformANTS would return the top-N (N specified in the request) Google-recommended documents as a function of distance between their document InfoPacks and the InfoPack of the specified user.

To measure how well the ordering resulting from the self-organizing arrangement process reflects the similarities communicated from the object model reasoners, we defined the Re-Rank Recommendation Overlap metric (short "Overlap"). This metric is applied when the response to a re-rank request is assembled. It consumes two ordered sequences of all InfoPacks other than the user model InfoPack of the request. In one ordering, the InfoPacks are sequenced by their distance to the user model InfoPack on the 3D torus (re-ranked ordering). In the other ordering the InfoPacks are sequenced by their respective similarity to the user model InfoPack as computed by an object model reasoner (the basis of the forces). The metric assumes that the InfoPacks in these two sequences are identical.

The overlap metric determines the percentage of InfoPacks that are the same for the top-N elements of the two sequences. N can range from 1 (only the closest or most similar InfoPack) to the full length of the two sequences. Since both sequence orderings are based on the same set of InfoPacks, the overlap for the full length of the sequences must be 100%. Two identical sequences have an overlap of 100% for any top-N subset, but any other pairs of sequences are not necessarily monotonic but trending towards 100% as more and more elements are included. Two random sequences have an expected overlap of N/M, where N is the size of the top-N subset and M is the total length of the sequences.

A typical cutoff (top-N value) for re-ranking requests in NIST evaluation was 12 out of 50 Google-recommended documents. Therefore, for the tuning of our system, it was most interesting to look at how much the top-12 InfoPacks derived from the arrangement in the locator overlaps with the top-12 most similar InfoPacks from the object model reasoner.

2.1.4.2.5.3.2 OVERLAP EXPERIMENTS

We used the Re-Rank Recommendation Overlap metric in preparation of the NIST evaluation experiment to

a) down-select from the set of five candidate object modeling approaches to those where the locator performs reasonably well in replicating the reasoner's similarity ordering, and then

b) tune the various parameters of the IMS for optimal ordering performance.

Table 3. Re-Rank Overlap for all Requests for User "himalia4" for all Object Modeling Approaches. Each x-axis indicates the top-N InfoPacks that are compared in the overlap metric (y-axis).

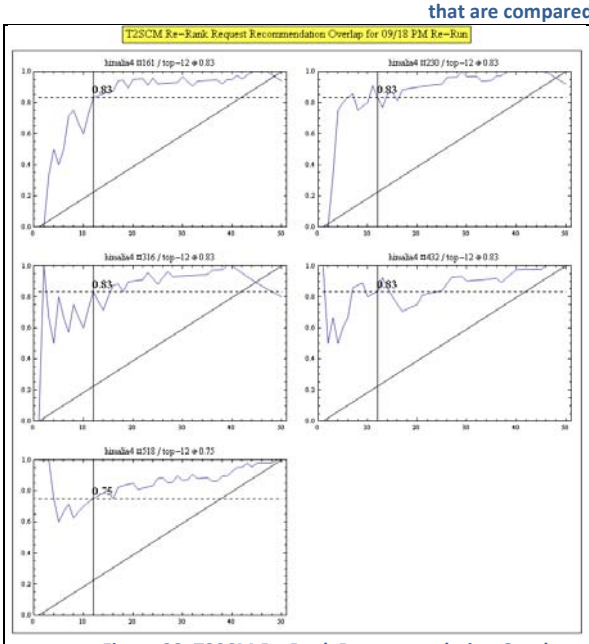

Figure 28. T2SCM Re-Rank Recommendation Overlap.

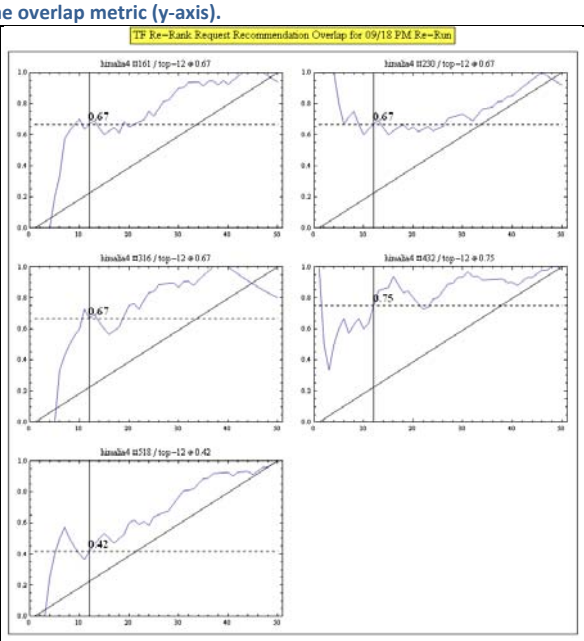

Figure 29. TF Re-Rank Recommendation Overlap

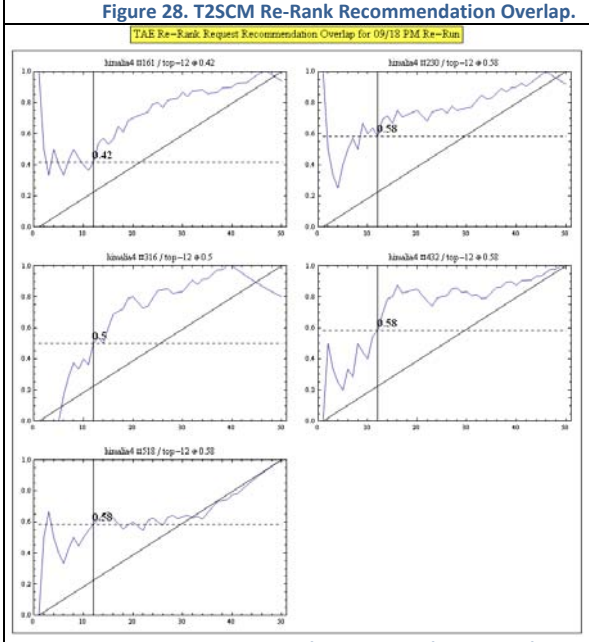

Figure 30. TAE Re-Rank Recommendation Overlap.

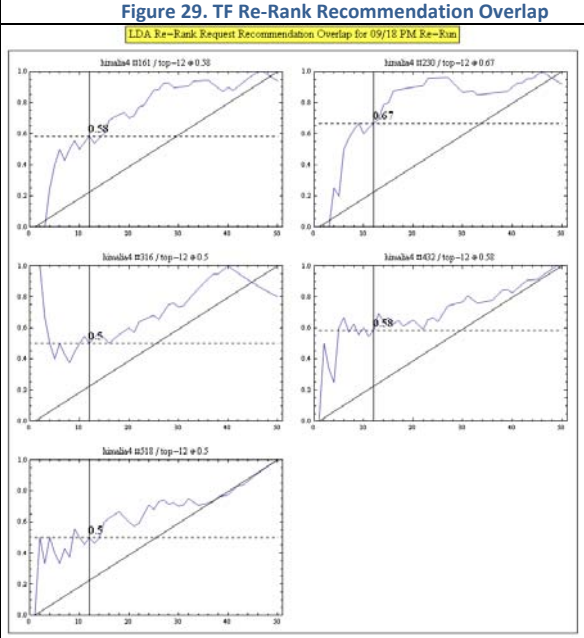

Figure 31. LDA Re-Rank Recommendation Overlap.

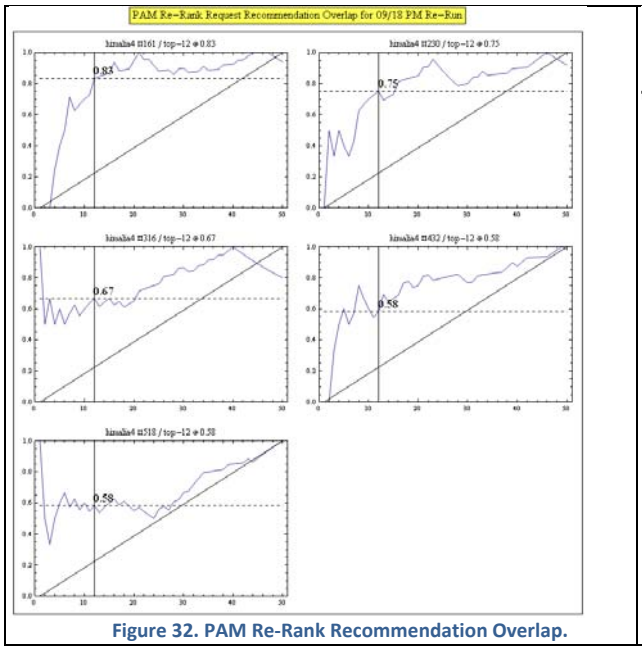

Figure 32. PAM Re-Rank Recommendation Overlap.

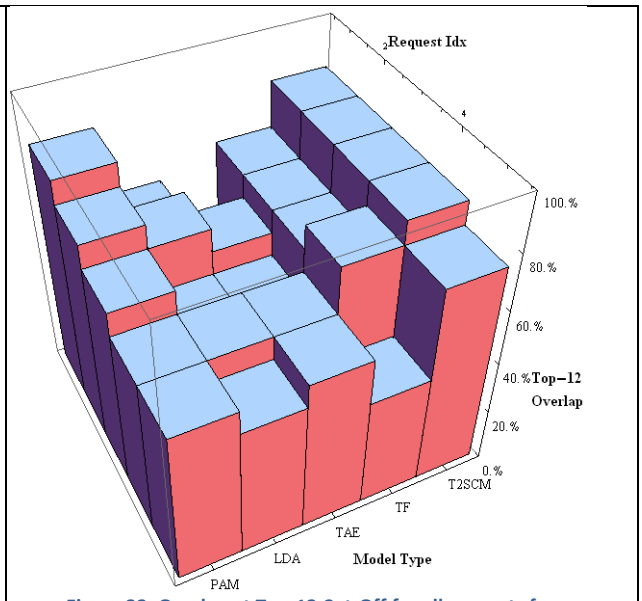

Figure 33. Overlap at Top-12 Cut-Off for all requests from user himalia4 (sequenced in "Request Idx" axis) and all Model Types.

Table 3 shows the overlap measures for the user "himalia4" from the NIST evaluation experiment in the afternoon of September 18, 2008. The plots show the measure applied to all five re-rank requests processed in this session and the data was obtained by re-running recordings of this session with each of the five object modeling approaches. The live experiment was performed with the TF model type because at the time significantly better user models were produced with this type, but the plots show that the self-organizing process in the IMS has similar performance for all model types.

2.1.4.2.6 LOCATOR GUI

To track the progress of the self-organizing InfoPacks in the locator and to control various parameters at runtime, we developed a graphical user interface for the locator component (Figure 34). The main control elements (left hand side) start, stop or reset the InfoPack agents that move according to the forces communicated by the reasoners. Also displayed on the left side are the recent messages (e.g., forces from reasoners, InfoPacks from UMS or IOMS) that were received by the locator.

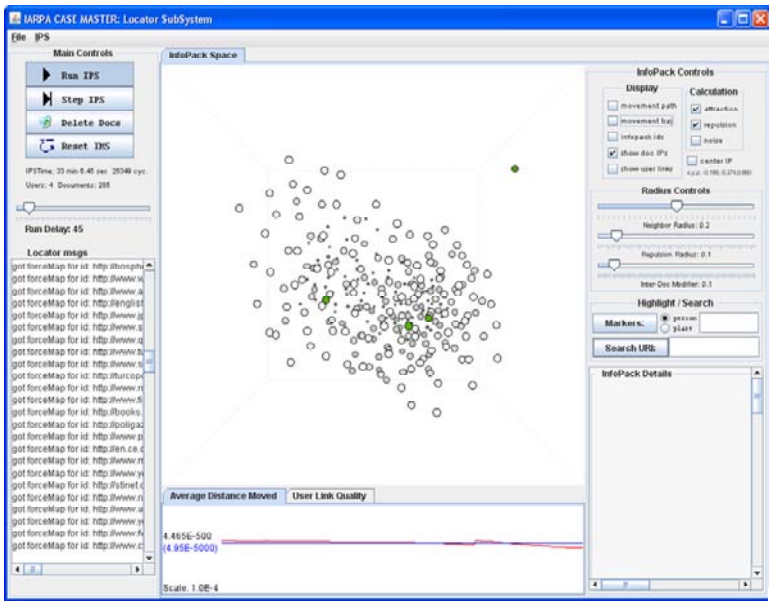

Figure 34. InfoPack Locator GUI displays and controls the self-organizing user (green) and document (gray) InfoPacks.

The InfoPacks self-organize into an appropriate arrangement in our information space that is a 3D torus. More specifically, the space is a unit-length box (1-by-1-by-1 in x/y/z) where the sides are "wrapped" around. So, in our GUI, we display the InfoPack locations inside this box and if InfoPacks cross the box boundaries, the InfoPacks are appropriately wrapped to the corresponding side. The view on the InfoPacks is a central projection of their 3D coordinates into our 2D window and we implemented drag-and-drop capabilities that provide simple maneuvering of the "camera" on the torus.

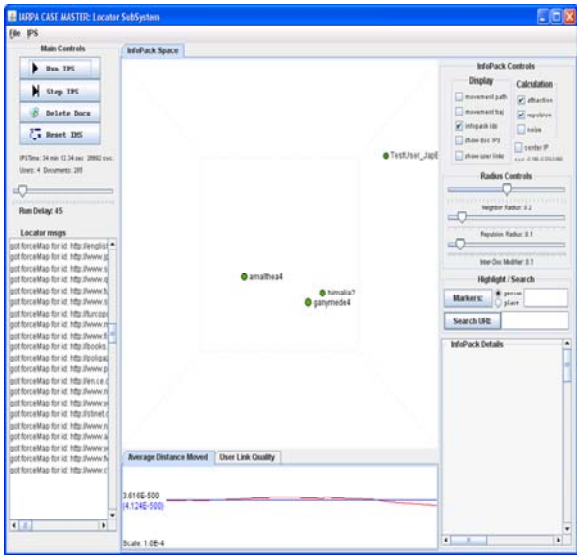

Figure 35. Display control shows only user InfoPacks.

The control elements on the right side of the window affect the way the InfoPacks are displayed in the center view ("Display" controls) and, more importantly, set various parameters of the self-organizing process for manual tuning of the prototype. We also have simple search mechanisms on that side that highlight InfoPacks by ID or by the SCM markers they carry. If a single InfoPack is selected by mouse-click in the central screen, its markers, its forces, and its neighbors are displayed in the "Details" window in the lower right. Figure 35 shows how only user model InfoPacks and their ID are displayed using the control elements.

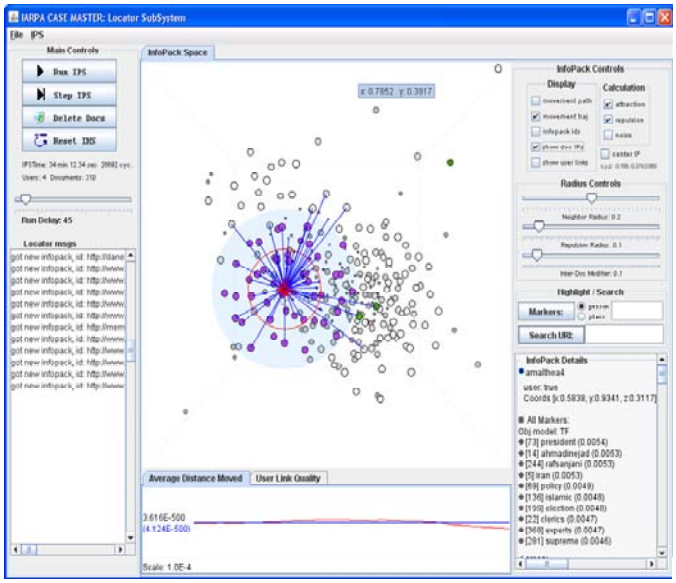

Figure 36. Show the forces that affect an InfoPack and show its neighbors within a fixed radius on the torus.

The tabs at the bottom of the window display real-time metrics that are calculated by the locator on the fly. Most importantly, the "Average Distance Moved" metric measures how much the InfoPacks are still moving in each step. Once the movement ceases (discount minute trembles due to rounding of very small numbers) the InfoPacks have found a force equilibrium.

The InfoPack movements are guided by forces to (attractive, set selectively by reasoners) or away from (repulsive, set indiscriminately by locator) other InfoPacks. The resulting trajectory may be very complex as it reflects the non-linear interactions of many autonomous components. Using the control elements of the GUI, we can explore the force patterns that guide a selected InfoPack visually and follow its trajectory over time. We can also view the collection of neighboring InfoPacks that would be included in the response to a getNeighbors call. See Figure 36 for an example screen shot.

2.1.4.3 IMS COMPONENT: DATABASE

The Database component of the IMS has three principal functions:

1. storing raw InfoPacks in memory
2. trace logging of all user InfoPacks
3. logging all VIGEstimates (from UMS and IMS separately) and user snap shots as xml files for post-process analysis

As new InfoPacks come into the IMS, the Database stores them in memory and makes them available to other components. For example, the Locator can request, by ID, an InfoPack from the Database. This minimizes memory overhead, as for most InfoPacks, only the Database stores the complete set of object models and markers for a given InfoPack.

Also, if a new InfoPack received by the Database is a user InfoPack, and user trace logging is enabled, the Database saves the user InfoPack as a raw XML file, so all user activity in the IMS can be tracked and later analyzed.

The Database also logs all VIGEstimates using the following procedure:

1. the database receives, asynchronously, a set of VIGEstimates (one for each model type) from the UMS
2. the database then saves each VIGEstimate to file
3. the state of all user InfoPacks is also saved at this time
4. next, the database makes a request (blocking call) to the Locator component, for an IMS VIGEstimate based upon the Locator arrangement algorithm
5. the database saves the Locator VIGEstimate to file
6. the database saves the state of all user InfoPacks at this time to file

2.2 VIRTUAL INTEREST GROUP (VIG) IDENTIFICATION

In InformANTS, we are exploring the use of the models described to identify virtual interest groups (VIGs) which are virtual communities of users with shared interests. InformANTS can help analysts collaborate by identifying their VIGs. By collaborating with other analysts, analysts can share information more quickly and more importantly avoid cognitive biases as a result of exposing to alternative opinions.

This VIG identification process works as follows. The UMS builds dynamic user models. UMS will be able to sort them in order of similarity by comparing their models in various ways. The most similar users will form a VIG. We have developed several applicable algorithms for identifying VIGs in the context of information exploration on the part of analysts.

We note that the self-organization of user-model InfoPacks in the Information Matching System (IMS, section 2.1.4) also results in the formation of virtual interest groups and we have put mechanisms in place in the most recent prototype, to log these VIG estimates in parallel to the UMS estimates for future comparisons and trade-offs. Synergizing the top-down VIG estimate approach (in the UMS) with the bottom-up approach (in the IMS) was planned for the third year of the (now curtailed) project.

2.2.1 VIG IDENTIFICATION ALGORITHMS

Formally, we define the problem of identifying VIG as follows. Given a user *u*, identify *k* system users that are most similar to u in terms of interests. The **Model-based algorithm** for VIG identification directly compares user models existing in terms of their component concepts and relations. The **Lucene-based[7] algorithm** casts the problem as an information retrieval task by transforming user models into textual documents and creating a TFIDF-based index. The **Tag-based algorithm** is related to social bookmarking applications such as del.icio.us and GDAIS (General Dynamics Advanced Information Systems, Inc.) Tag|connect. This algorithm judges similar users by their tagging activities.

2.2.1.1 MODEL-BASED ALGORITHM

Assume that UMS has adapted interest models for a large number of users over some period of time. The pseudocode for the algorithm is as follows.

Compare the similarity of u's model with that of another system user x

Two concepts or relations are similar if

1) their definitions share information content, i.e. terms.

2) their relative weights in their own model are similar

The number of concepts and relations that are similar between the compared user models

If the similarity exceeds a preset threshold, add x to the VIG for u

If the size of VIG for u is equal to k, stop

2.2.2 LUCENE-BASED ALGORITHM

This algorithm first transforms each user model in the system to a textual document and then indexes it using Lucene. Next, we transform the given user model into a query. We then search the query in the index and return the top N documents that are most similar to the query in terms of TFIDF as the VIG. The score of each top document is the similarity to the given model. The index needs to be updated when a user model is updated. This is done by deleting the document representing the model and re-indexing the corresponding document.

[7] Apache Lucene is a high-performance, full-featured text search engine library written entirely in Java.

2.2.2.1 CFA (COLLABORATIVE FILTERING INSPIRED ALGORITHM)

This algorithm is based on the collaborative filtering framework, treating user model elements (i.e. concepts and relations) as "items" and building a matrix of users vs. model elements (Figure 37). The value for each cell is the interest weight of the corresponding element for the corresponding user. CFA involves the following three steps:

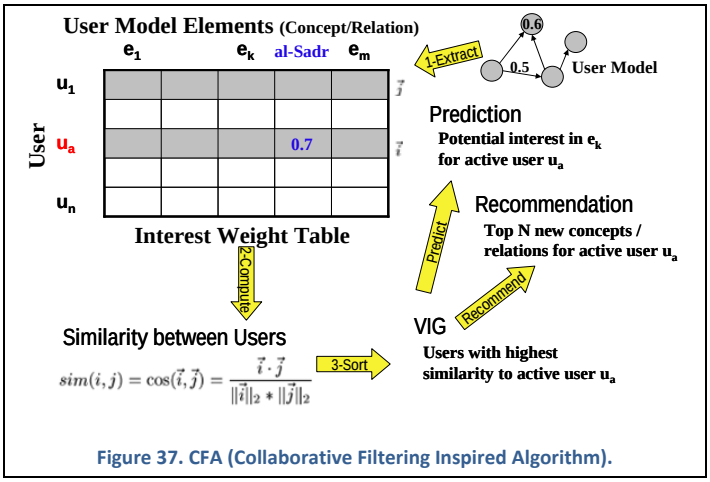

Figure 37. CFA (Collaborative Filtering Inspired Algorithm).

1) Extract concept and relation weights from the user model to build the interest weight table. Assume that the UMS has created a user model for each system user. For each user, enter all element weights into the user-element matrix. If a user does not have an element in her model, the corresponding cell value will be 0.

2) Compute cosine similarity between the active user and each system user.

3) Sort the system users based on similarity to form VIG for the active user. The users with highest similarity form the VIG.

Once we identified a VIG for the active user, we can go a step further to 1) recommend top N new concepts or relations to the user; and 2) predict the active user's potential interest in a new concept or relation that is not part of her user model.

2.2.2.2 TAGA (TAG-BASED ALGORITHM)

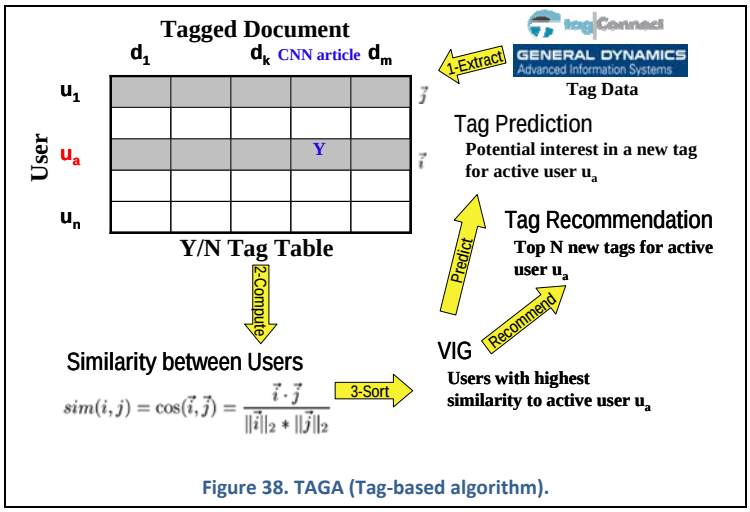

Figure 38. TAGA (Tag-based algorithm).

This algorithm assumes that we have captured tagging activities of users using a social bookmarking application such as GD AIS's tag|Connect. This algorithm also relies on the collaborative filtering framework (Figure 38). The main difference from CFA is that TAGA does not require the system to build user models first. Here we build a user-tagged document matrix and populate a Y/N tag table. Each cell is filled with a Y if the corresponding document was tagged by the corresponding user and an N otherwise. Once we have the tag table populated, we compute the similarity between the active user and every system user. The users with highest similarity form the VIG.

The VIG and the tag table also allow the system to make tag recommendations and predictions straightforwardly. When the active user encounters a new document, the system can find the tags that her VIG members use for the same document and recommend. For tag prediction, the system can compute the frequency with which VIG members use it for the target document.

2.2.3 VIG DEMO

One of the main goals for UMS is to identify Virtual Interest Groups (VIGs), which helps analysts to find colleagues with similar interests to collaborate. In our last monthly, we described three algorithms for identifying VIGs. One such algorithm is the **Lucene-based[8] algorithm**, which casts the problem as an information retrieval task by transforming user models into textual documents and creating a TFIDF-based index. We have recently implemented this algorithm in the UMS VIG demo we presented at the last CASE PI meeting.

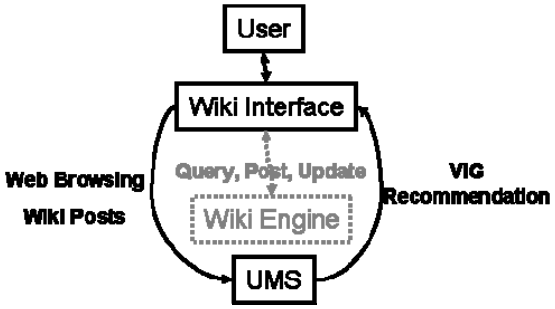

Figure 39. UMS VIG demo architecture.

The architecture of the demo is shown in Figure 39. The user interacts with a Wiki Interface, which is a custom Web browser where the user can surf the Web, including the Wikipedia Web site. All pages that the user surfs to are being tracked and used to build a user model by the UMS. On the other hand, UMS created user models for 437 registered Wikipedia users offline. For each one of these users, UMS has retrieved the user's home page and 20 most recent posts by the user from the Wikipedia Web site. This user information was used to create the model for the user. For this demo, as the active user surfs the Web (Figure 40, left), the user's model is continually being updated. The active user can also view her own user model from the interface at any time (Figure 40, middle). The VIG is computed using the Lucene-based algorithm in an on-demand manner. The VIG is displayed as a list of 20 most similar users with the similarity shown (Figure 40, right). The user views the VIG at any time. In addition, the user can explore the members in the VIG. Whenever a member is clicked, the member's user model is being shown as a tree structure. A Web page containing links to the member's home page and its contributions (i.e. posts) is also shown.

[8] Apache Lucene is a high-performance, full-featured text search engine library written entirely in Java.

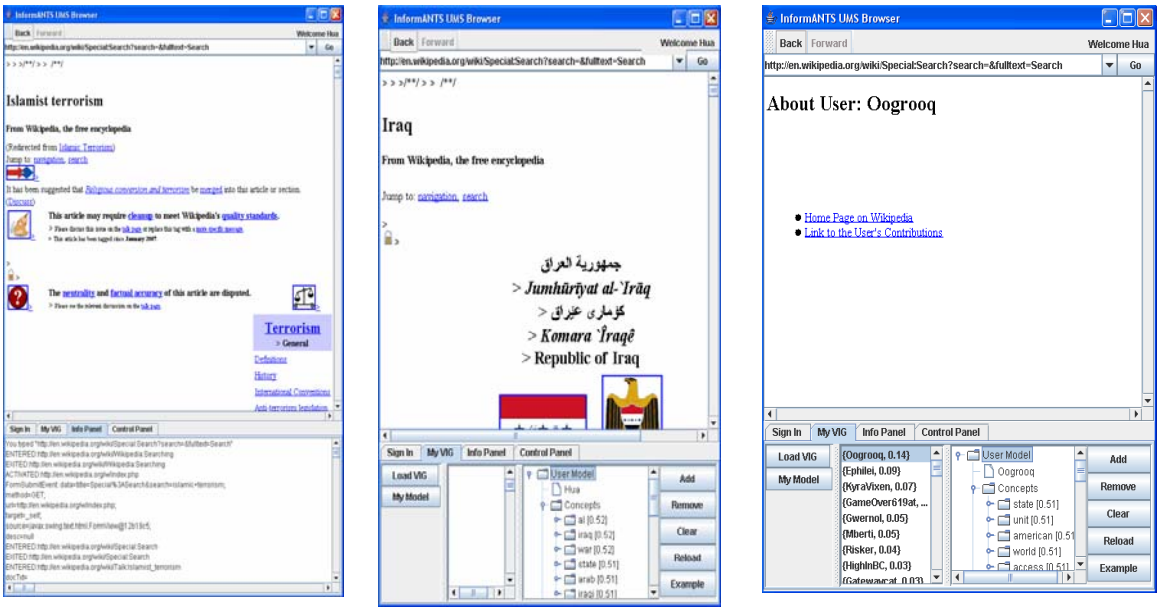

Figure 40. UMS VIG demo screen shots showing the active user surfing (left), active user model (middle), and the VIG and the highlighted member info (right).

2.2.4 VIG EXPERIMENTS

2.2.4.1 TAG PREDICTION USING TAGA

We have investigated Collaborative Filtering for the tasks of VIG identification and tag prediction in the TAGA algorithm. Collaborative Filtering is a collection of well established techniques for predicting user behavior based on the behavior of other users. It has proved especially successful in the online retail world where millions of users rate or buy from inventories of unprecedented size. There are two major types of Collaborative Filtering algorithms: user-based and item-based. We have focused on user-based, where similarity metrics are used to find the correlation between each pair of users and a neighborhood of users (~VIG) near each individual user is built according to the correlations. From a weighting of the interests of a neighborhood's users, the algorithm predicts individual user interest in items which the neighborhood has seen, even though the individual user has not seen them.

To begin evaluating the value of Collaborative Filtering, we applied the open-source Collaborative Filtering toolkit *Taste* to the General Dynamics' tag|Connect database of user-tagged URLs. In the initial attempts, we did no better than chance. However, when dropping all tags which a user has only applied once or twice, we were able to predict 20% of the held out tags, which is significantly better than chance. (These experiments are preliminary and still require validation.)

2.2.4.2 ANALYST TASKING IDENTIFICATION

This is an experiment where the VIG algorithm is used to identify the analysts that are working on the same or similar taskings by processing the analytic events (query, page view, cut and paste, etc.). We conducted such an experiment for the user modeling challenge.

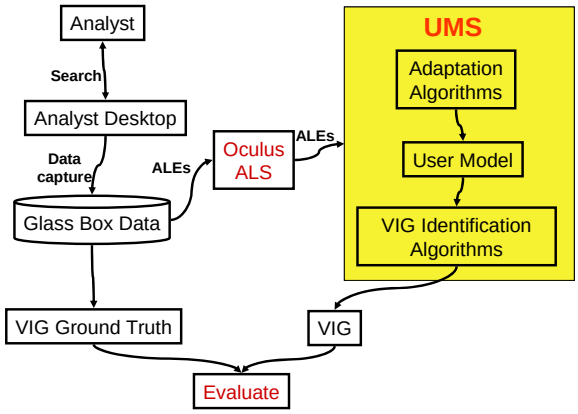

Figure 41. UMS modeling challenge architecture.

The challenge problem is: Can the UMS identify virtual interest groups (VIGs) via user modeling? We have developed an experimental approach to address this problem. The architecture for this approach is shown in Figure 41. At a high level, in the experiment we build user models using the NIST Glass Box data as input and then use them to identify VIG for each user. We then compare the VIG with the ground truth that NIST holds. We have conducted extensive experiments and were able to show very promising results in identifying clusters of individuals with similar analytic roles. We provide a detailed summary of the results in this section.

2.2.5 THE EXPERIMENT WORKFLOW

The workflow for the experiment is as follows.

- Install Glass Box Data Release 421
- Establish ground truth: Historical users with the same tasking form a VIG
- (NIST) - Extract analytic events for all historical users from the Glass Box and store in ALS
- Build models of historical users with tasking information excluded in the modeling process
- Offline evaluation by SET (or NIST)
    - For each historical user, UMS finds a VIG
    - Evaluate results against ground truth

- **Ccgbuser4**
  - 100401-Syria Reaction
  - 11/14 – 11/23/2005
  - 588 ALEs
- **Ccgbuser7**
  - 100701-FSU Biotech Council
  - 11/14 – 11/22/2005
  - 1107 ALEs
- **lc1 – lc6**
  - lc week 2
  - 11/14 – 11/18/2005
  - ALEs: lc6=487, lc5=311, lc4=409, lc3=304, lc2=318, lc1=135
- **ltrist1 – ltrist6**
  - Tasking unknown
  - Ltrist1 9/28-9/30/05; ltrist2, 4, 5: 9/26-9/30/05; ltrist3: 9/26/05; ltrist6: 9/27-9/30/05
  - ALEs: ltrist6=828, ltrist5=1002, ltrist4=2612, ltrist3=325, ltrist2=1262, ltrist1=516

Figure 42. The NIST data breakdown.

2.2.6 THE NIST DATA SET

The data we used for the tests are the ALEs hosted by NIST ALS. The ALEs are converted from the Glass Box Data cycle 6 and cycle 10 collected by NIST in 2005. There are a total of 14 users. Their tasking, working dates and number of available ALEs are shown in Figure 42.

2.2.7 EXPERIMENTAL DESIGN

For each user, we split the available ALEs into N parts, where N is a parameter (Figure 43). We build a model for each individual part. With M users, where M is another parameter, we shall have N*M models. Next, we take each model and try to find 5 models out of the N*M models (exclude the model itself) that are most similar.

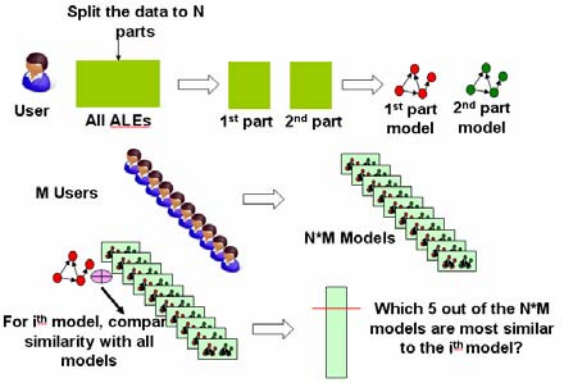

Figure 43. Experimental design.

2.2.8 TEST WITH FULL-DAY MODELS

For this test, we use 8 users: ccgbuser4 and 7, and lc1 - lc6 (Figure 44). We split the data into daily parts. We then build one model fresh per day for each user. For example, the model ccgbuser4.11_14GB represents the model on day Nov. 14, 2005 for user ccgbuser4. Together we have a total of 38 full day models. For each model, a VIG of 5 members is identified. Thus we have a total of 38*5=190 VIG members.

	cg4	cg7	lc1	lc2	lc3	lc4	lc5	lc6
	ccgbuser4.11_14GB	ccgbuser7.11_14GB	lc1.11_14GB	lc2.11_14GB	lc3.11_14GB	lc4.11_14GB	lc5.11_14GB	lc6.11_14GB
	ccgbuser4.11_15GB	ccgbuser7.11_15GB	lc1.11_15GB	lc2.11_15GB	lc3.11_15GB	lc4.11_15GB	lc5.11_15GB	lc6.11_15GB
	ccgbuser4.11_16GB	ccgbuser7.11_16GB	lc1.11_16GB	lc2.11_16GB	lc3.11_16GB	lc4.11_16GB	lc5.11_16GB	lc6.11_16GB
	ccgbuser4.11_17GB	ccgbuser7.11_17GB	lc1.11_17GB	lc2.11_17GB	lc3.11_17GB	lc4.11_17GB	lc5.11_17GB	lc6.11_17GB
	ccgbuser4.11_18GB	ccgbuser7.11_18GB						lc6.11_18GB
	ccgbuser4.11_22GB	ccgbuser7.11_21GB						
	ccgbuser4.11_23GB							
Total Models	7	6	4	4	4	4	4	5
Total VIG Members (x 5)	35	30	20	20	20	20	20	25

Figure 44. Built full-day models.

	cg4	lc1	lc2	lc4	lc6	cg7	lc3	lc5
cg4	10	3	4	6	9			
lc1	2	5	2	1				
lc2	12	4	6	7	7			
lc4	6	5	4	4	5			
lc6	5	3	4	2	4			
cg7						12	5	5
lc3						10	7	7
lc5						8	8	8

Figure 45. Results with full-day models. Precision=100%.

The results with full-day models are shown in Figure 45. Each column shows the VIG membership profile for the user identified by the column title. For example, the column labeled as cg4 is for ccgbuser4. We know from Figure 44 that this user has 7 full-day models with 35 associated VIG members. Of the 35 members, 10 originated from the user herself, 2 from lc1, 12 from lc2, 6 from lc4, and 5 from lc6. None of the members come from cg7, lc3, or lc5. The red numbers in the diagonal indicate the membership contribution from the same user. The columns are arranged such that ones with similar membership profile are moved together. It is clear from the figure that the users are separated into two clusters. One is formed by ccgbuser4, lc1, lc2, lc4, and lc6 while the other by ccgbuser7, lc3 and lc5. This result matches perfectly the ground truth from NIST.

2.2.9 TEST WITH HALF-DAY MODELS

For this test, we used 14 users: ccgbuser4, 7, lc1 – 6, and ltrist 1 – 6. Each day is split in half. One model is built fresh for each half for each user. A total of 106 half-day models are built. Again for each model, a 5-member VIG is identified. Thus there are 106*5 = 530 VIG members altogether. The VIG membership distribution is shown in Figure 46. The arrangement of the columns is similar to that in Figure 45.

The ground truth is indicated by fill color of the cells: green=positive match, white=negative match, pink=mismatch, and grey=miss. A positive match means our system correctly identifies that the user in question and the VIG membership contributor belong to the same group. On the other hand, a negative match means our system correctly identifies that the two users do NOT belong to the same group. We computed the precision using the following formula:

Precision = (positive match + negative match) / (all cells) = 87.2%

	cg4	lc1	lc2	lc4	lc6	cg7	lc3	lc5	lt1	lt6	lt2	lt5	lt3	lt4
cg4	18	1	9	5	11	1						3		
lc1	7	16	5	6	2					1				
lc2	15	6	17	11	11									
lc4	9	7	7	13	7	1								
lc6	15	5	7	5	19	2		1				2		
cg7						23	7	3	1			1		
lc3						8	20	12	2	2				
lc5						12	13	22	2	1		1		
lt1						1	2	1	8					1
lt6						2	3		1	16	2	1	9	8
lt2										1	26	8		1
lt5	1					4		1		9	20	33		2
lt3									2	6			4	7
lt4						1			4	4	2	1	2	31

Figure 46. Results with half-day models. Precision=87.2%. Ground truth shown in cell colors: green=positive match, white=negative match, pink=mismatch, grey=miss.

2.2.10 FIREFOX EXTENSION FOR USER ACTIVITY LOGGING

A Firefox extension is composed of scripts, xul GUI layout files, java-jar files, and style sheets all installed as a zipped folder in a user's profile. Advantages are portability, easy access to a fully developed browser application, and quick UI prototyping. As a development and demo tool, this can provide easy extensibility for additional logging, in addition to an appealing interface that can interact with the user using minimal screen real estate and no context switching. Because Firefox is built on the Mozilla Component Framework and exposes the Document Object Model for content, essentially any behavior can be monitored and any content observed. In the spirit of the popular GreaseMonkey tool, even specific web applications such as Gmail (AJAX) could be monitored. As a tool with potential for use by other CASE teams and actual analysts, we can provide logging for a commonly used browser, almost no training is required, and, in theory, it would be easier to get an extension accepted by a community concerned with security than a new application.

Currently, we can track:

- *Search Queries*. Not only from within the tool, but also queries from google.com. Note that behavior in forms can be observed, though here we watch for the query embedded in the link.
- *Page loads*. What links does the user follow?
- *Most other kinds of content loads* (e.g. frames in the page). Important because browsing is no longer solely a page-by-page activity.
- *Copy/Paste*.

The user's actions are logged within the tool.

2.3 VIGOR (VIG OVERSIGHT AND REPAIR)

One of the services provided by InformANTS, as well as by other CASE efforts, is the identification of virtual interest groups (VIG's). The motivation for such a service is that if analysts can learn more readily about colleagues who share their interests and concerns, they will be able to leverage one another's expertise and produce higher-quality products more efficiently.

In addition to such beneficial effects, VIG identification might pose some challenges for which we should be prepared. Analysts have only a finite amount of time to spend interacting with one another. If VIG's enable them to interact more with people pre-screened to share their interests and concerns, they will have less time to interact with people of different profiles. This might conceivably lead to a narrowing of their perspective due to positive feedback within VIG's, a process we term "communal cognitive convergence" or C^3. If unchecked, C^3 might actually lead analysts who make extensive use of VIG's to produce lower quality products. VIG's might actually funnel their thinking into narrow stovepipes, cause them to overlook key data or concepts outside the group's attention, lead to premature rejection of competing hypotheses.

During this period, we conducted a modest effort to explore the potential for C^3. Here we report on two lines of evidence for this phenomenon and suggest implications for the CASE program.

2.3.1 EXPERIMENTAL EVIDENCE

Our initial approach to C^3 was to see if we could construct a simple computational model [8-10] that captured the dynamics.

We instantiate a population of N agents, each with a binary vector $I = \{0,1\}^L$ of interests. If $I[j] = 1$, the agent is interested in topic j, while a 0 means it is not interested in that topic. Initially, the strings are randomly generated with the probability of each topic being 0.5.

We can measure the distance between the interests I of any two agents, using any of a number of distance measures in [0,1] between binary vectors (e.g., normalized vector angle, Jaccard measure, Dice measure, Yule measure, normalized Hamming distance). We define an agent's *community* as the set of all agents whose interests fall within a threshold θ of its own. The community's *focus* on a topic j is the proportion of agents in the community whose interests have a '1' at position j.

Within this structure, we execute two processes, capturing simple intuitions about learning and forgetting.

The intuition about learning is that an agent is likely to gain an interest in a topic of many of the individuals in its community that are interested in that topic. To model this intuition, with probability p_{learn}, an agent randomly selects one of the bits j in its interest vector. If the bit is 0, it flips it to 1 with probability cf(j), where cf(j) is its community focus on bit j.

The intuition about forgetting is that an agent is less likely to forget a topic if its community shares its interest in that topic, and more likely to forget it if the community isn't interested in it. To model this intuition, with probability p_{forget}, an agent again randomly selects a bit j in its interest vector. If the bit is 1, it flips it to 0 with probability 1 – cf(j).

We explored this model with N = 20, L = 10, θ = 0.5 , $p_{learn} = p_{forget}$ = 0.9. The actual rate of learning or forgetting is much lower than these values would suggest, for two reasons. First, at each step, the agent chooses with equal probability whether to learn or to forget, so on average each process executes only half of the time. Second, the process makes no change to the agent if it is forgetting and the chosen bit is already 0, or if it is learning and the chosen bit is already 1.

We run the model for 1000 steps, and wish to track how the similarity among similar agents changes in comparison to the similarity among dissimilar agents. A convenient way to observe the structure of the population is to cluster the agent's interests hierarchically, as shown in Figure 47. In Generation 0, given random interests, the separations at which individual agents join into clusters are fairly large compared with the separations at which clusters join into larger clusters. But in Generation 1000, most agents are part of one of two clusters with 0 separation, while the largest separation (about 0.7) is larger than at the start of the process.

We can formalize this observation as a measure of population diversity. For a given population, cluster the population hierarchically, and measure the separation at which each agent first joins a cluster. Take the median of these separations, and divide by the diameter of the global cluster. The result, the MinMax Ratio, is a number between 0 and 1. When the number is small, agents are much closer to their nearby neighbors than they are to the most distant agents. When it is large, agents are more evenly separated from one another.

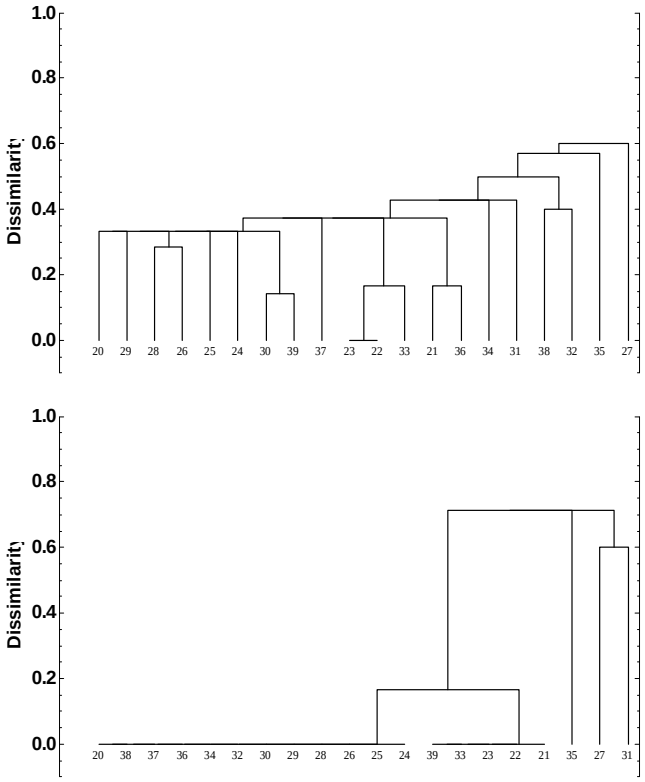

Figure 47: Population Structure.--Top: Generation 0. Bottom: Generation 1000.

Figure 48 plots the MinMax Ratio as a function of generation. It decreases steadily, until by generation 150 almost all agents are part of individual clusters with diameter 0, indicating that they have become identical with their neighbors.

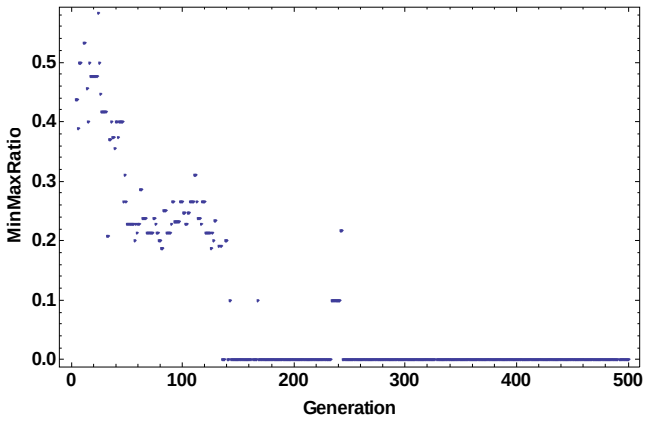

Figure 48: MinMax Ratio as function of Generation

A similar line of simulation research in the political science community confirms our results. The most prominent study is Robert Axelrod's Adaptive Culture Model (ACM) [1], in which agents situated on a lattice interact pairwise with their immediate neighbors based on how similar they are to one another. This model also shows the convergence of agents into groups. It differs from our model in that its interactions are all pairwise and do not take into account the effects of group size and degree of community focus.

2.3.2 HISTORICAL EVIDENCE

The C^3 effect has been documented empirically in a wide range of settings [6]. It is related to the runaway process observed by R.A. Fisher in evolutionary settings [3], in which a trait that offers no survival benefit under natural selection may become the focus of sexual selection, resulting in extravagant developments such as the peacock's tail as females select males with the trait and produce offspring that reinforce the preference (on the female side) and the trait (on the mail side).

2.3.3 THE WAY AHEAD

Recognition of the C^3 effect suggests that the CASE program should devote some effort to VIGOR (Virtual Interest Group Oversight and Repair). Such an effort should

- Articulate the risk of C^3 inherent in VIG tools
- Refine and extend the modeling effort described above to understand the phenomenon more fully, including the effects of spatial separation or co-location and multiple group membership and the dynamics of convergence
- Develop mechanisms to detect and monitor C^3
- Develop techniques to counter C3, such as limiting group lifetime or seeding groups with members orthogonal to the interests of the main body.

2.4 MUTUAL INFORMATION AND MARKER DIMENSIONALITY

We found that document markers produced by generative topic modeling (GTM), whether the topics themselves or the keywords underlying them, yielded lower mutual information (MI) with respect to the document population than did keywords produced by Lucene or specialized concepts produced by InformANTS' T2SCM tool. In that study, based on 699 Apex documents, the dimensionalities of the Lucene and T2SCM marker sets were much larger (>4k and >10k, respectively) than those of the GTM marker sets (30 or 100 for topics, 543 or 1770 for derived keywords), and it was suggested that the difference in MI might reflect the difference in dimensionality. We have computed MI for marker sets derived from Lucene and T2SCM but constrained to lower dimensionality, and find that dimensionality does not explain the observed difference in MI. The same trends apply to PAM-derived topics as well as to those derived by LDA, and are seen in the Monterey data as well as the Apex data.

Table 4. Monterey and APEX Data

Monterey Data					APEX Data				
	# Docs	# Unique KWs	KWs/ Doc	MI		# Docs	# Unique KWs	KWs/ Doc	MI
LDA (100-Topic) Topics	1389	100	0.07	0.147	LDA (30-Topic) Topics	699	30	0.04	0.092
LDA (100-Topic) Keywords	1389	1775	1.44	0.13	LDA (30-Topic) Keywords	699	543	0.86	0.084
PAM (100-Topic) Topics	1389	100	0.07	0.163	PAM (30-Topic) Topics	699	30	0.04	0.07
PAM (100-Topic) Keywords	1389	1296	1.44	0.124	PAM (30-Topic) Keywords	699	448	0.86	0.056
T2SCM	1389	10119	31.07	0.504	LDA (100-Topic) Topics	699	100	0.14	0.175
T2SCM (restricted)	1389	8479	27.17	0.481	LDA (100-Topic) Keywords	699	1770	2.86	0.154
Lucene	1389	4302	31.08	0.463	T2SCM	699	10401	42.15	0.566
T2SCM (10 KWs/Doc)	1389	2844	9.48	0.481	T2SCM (restricted)	699	8779	37.27	0.55
T2SCM (10, restricted)	1389	2628	9.36	0.458	Lucene	699	4039	42.16	0.473
Lucene (10 KWs/Doc)	1389	1853	9.49	0.502	T2SCM (10 KWs/Doc)	699	1953	9.54	0.541
T2SCM (3 KWs/Doc)	1389	798	2.94	0.46	T2SCM (10, restricted)	699	1892	9.41	0.529
T2SCM (3, restricted)	1389	734	2.93	0.441	Lucene (10 KWs/Doc)	699	1355	9.55	0.53
Lucene (3 KWs/Doc)	1389	735	2.95	0.531	T2SCM (3 KWs/Doc)	699	590	2.94	0.533
T2SCM (2 KWs/Doc)	1389	505	1.97	0.455	T2SCM (3, restricted)	699	564	2.94	0.521
T2SCM (2, restricted)	1389	466	1.96	0.438	Lucene (3 KWs/Doc)	699	526	2.95	0.562
Lucene (2 KWs/Doc)	1389	530	1.98	0.535	T2SCM (2 KWs/Doc)	699	401	1.97	0.522
T2SCM (1 KWs/Doc)	1389	252	0.99	0.459	T2SCM (2, restricted)	699	383	1.97	0.507
T2SCM (1, restricted)	1389	227	0.98	0.422	Lucene (2 KWs/Doc)	699	371	1.98	0.566
Lucene (1 KWs/Doc)	1389	273	1	0.547	T2SCM (1 KWs/Doc)	699	196	0.99	0.486
					T2SCM (1, restricted)	699	185	0.99	0.462
					Lucene (1 KWs/Doc)	699	212	1	0.581

2.4.1 REDUCING KEYWORD DIMENSIONALITY

We restricted the number of keywords that we get from the non-GTM mechanisms (Lucene, and T2SCM) by taking only the most common n keywords from each document, for n in {1, 2, 3, 10}. As a result, the total number of unique keywords drops from the original high values (10119 for T2SCM, 4301 for Lucene) down to fewer than 300, within the range of dimensionality sampled by the GTM methods. In addition, our experiments with the IMS have showed us that it is important to use only specialized concepts for which characteristics (e.g., age of people, lat/long of places) are properly supplied, so we computed the MI for T2SCM specialized concepts restricted in this way. At BAE's suggestion, we also analyzed 1389 documents from the Monterey data, and included topics and topic-derived keywords generated by PAM as well as LDA. Table 4 shows the results for the Monterey and the Apex data. Figure 49 summarizes these results graphically. The two points for each GTM series are for the topics (the smaller set) and the associated keywords (the larger one).

This analysis shows that the higher MI with Lucene and Specialized Concepts (SC) is not the result of higher dimensionality. Even with their dimensionality restricted to be less than the keywords derived from GTM methods, Lucene and SC still deliver much higher MI.

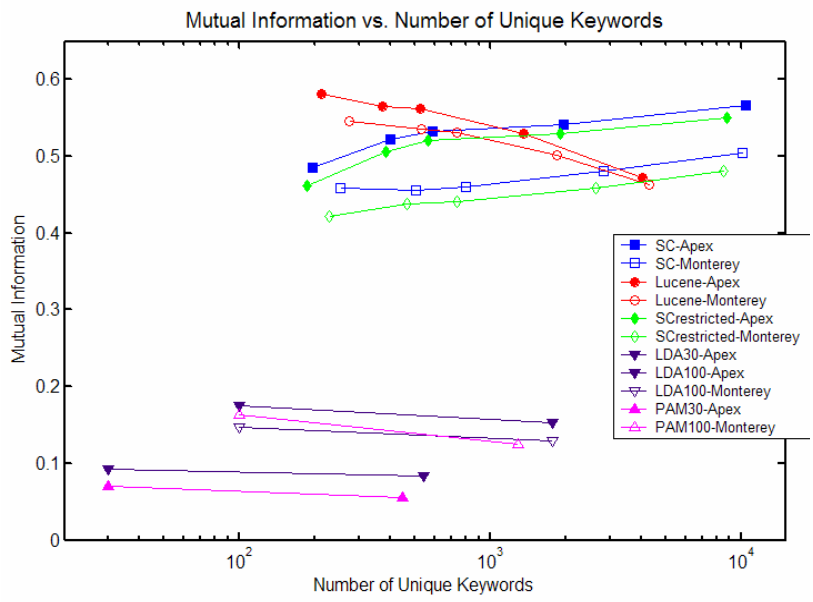

Figure 49: Summary of MI vs. Dimensionality

Incidentally, note the crossover between Lucene and SC. As the number of dimensions increases, Lucene-derived keywords actually yield lower MI, while the MI from specialized concepts increases (slowly). This effect can be explained by noting that additional keywords derived by T2SCM are restricted to terms with high likelihood of being diagnostic of individual documents (persons and places), while the rarer keywords from Lucene are less likely to be semantically characteristic of their documents.

In our earlier analysis, we noted that LDA generated topics with a very uneven distribution, as shown in Figure 50(left). We hypothesized that this uneven distribution was responsible for the lower MI when topics (and their associated keywords) were used as document markers. PAM topics are much more evenly balanced (Figure 50 right), but this does not yield appreciably better MI scores (Figure 49). This observation invalidates our previous explanation of the lower MI scores for GTM-generated markers, and leaves the question open.

2.5 VALUE OF INFORMATION

The InformANTS team participated in a number of discussions with other CASE participants on refining the concept of the Value of Information. The concept is still very much in flux, and our work on it has been curtailed in order to support the final experiment. To capture the key issues for future development, this section summarizes our discussions as of the April 2008 PI meeting in Orlando, then points to a number of other discussions in the literature that need to be brought into the discussion, and finally suggests a new perspective that may help to integrate these various ideas.

2.5.1 STATE OF THE DISCUSSION, APRIL 2008

Discussions between Van Parunak and Rob Hyland of BAE led to the notion that the various dimensions of the value of information relate to different entities: documents, users, and analytic events (including decisions). We sketched out the following set of dimensions:

- Relevance is semantic similarity between documents and users. It may relate Docs to Docs, Users to Users, or Docs to Users. The appropriate calculus is probably grounded in information geometry (e.g., Hellinger divergence).
- Novelty is a temporal measure of how long a User has been interacting with a Doc. This is a fairly touchy dimension; see below for more detail, where we will suggest that the appropriate calculus might be based on digital pheromones.
- Credibility is a measure of the Truth of the assertions in a Doc, so the appropriate calculus is some truth-valued logic that can reason about combination and propagation of evidence. Perhaps Dempster-Shafer theory or Bayes nets would serve. It might also be related to internal consistency among documents, or the reputation of the metadata associated with a document.

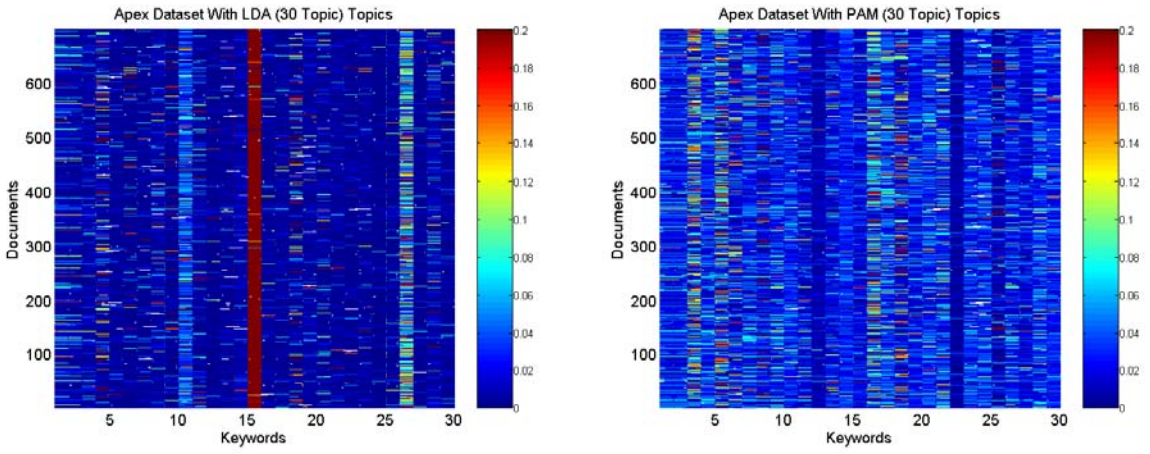

Figure 50: Distribution of topics derived via LDA (left) and PAM (right)

- Persuasiveness is a relation betwen Docs and Decisions, describing how much a Doc influences a given Decision. It is directed from Doc to resulting Decision. The appropriate calculus would probably need to include some psychological measure of the susceptibility of the decision-maker to the particular Doc. Further discussion suggested that persuasiveness involves diagnosticity (how much it separates rival hypotheses), and the mutual information between evidence and hypothesis or action. (Note that "hypothesis" is not well defined in our initial taxonomy of entities, but see below.)
- Completeness is how much a user knows about a topic. (Again, note that "topic" is not well defined in our taxonomy of entities.)

Now let's talk more about Novelty.

Novelty measures a relation between a User and a Doc: how recently has the User interacted with the Doc. We suggest the following features of novelty:

1. A Doc is highly novel if the User has never encountered it before. The more the user interacts with it, the less novel it becomes.
2. If a User interacts with a Doc, then abandons it for a while, on a later encounter it may appear Novel again. So we need to model the "forgetting process."
3. A User who interacts with a Doc based on one tasking will see it through the lens of that tasking; it might take on a totally different meaning when viewed through a different tasking. So we need to modulate novelty based on the user's perspective.

This time-based approach to Novelty is a shift from our previous discussions, which focused on semantic dissimilarity. But that approach seems less fruitful; it is essentially the complement of Relevance, and that is such a large class as not to be useful. We propose to define Novelty on the basis of time rather than semantic similarity to the user's interests, leaving Relevance to handle semantic similarity. Time might result from other more complex factors. For example, imagine that a document is known to someone in the social network of which the User is a part. The further someone acquainted with the Doc is from the User, the longer it will take the Doc (or news about it) to reach the user. So Novelty reflects, indirectly, how far from the User the Doc was in the social space.

Here's an operational way to implement Novelty, based on digital pheromones. The more of a User's pheromone is on a doc, the less novel the Doc is to that user.

- For simplicity, let's view each InfoPack as a distribution over topics. Novelty pheromone is distributed across the Doc's topics, but novelty is assessed based on the sum of the pheromone across the topics.
- At each time step, each User deposits digital pheromone on all Docs she's actively using or consulting. The deposit is allocated across the Doc's topics proportional to the product of the Doc's score on that topic and the User's score on the same topic. The idea is that we credit the Doc with non-Novelty only to the degree that it aligns with the User's current user model.
- At each time step, we evaporate some fraction of the pheromone from each Doc.

Note how this scheme achieves our desiderata for Novelty.

1. When a User first encounters a Doc, it has zero pheromone for that User, and so is highly novel. The longer the User interacts with the Doc, the more pheromone accumulates, reflecting decreased Novelty.
2. If a User does not interact with the Doc for a while, the pheromone evaporates, so that the Doc's novelty increases.
3. If a User approaches the Doc from a different perspective, reflected in a changed user model, the pheromone is credited to different topics, permitting a Doc to appear novel based on a shift in interest.

On further discussion, it was felt that there is a need to distinguish Newness from Novelty, to tease apart absolute from user-relative features.

2.5.2 OTHER DISCUSSIONS IN THE LITERATURE

There is an extensive discussion in the literature of "Information Quality" (IQ), which has many points of overlap with our discussions on Value of Information. A commonly cited reference [7] distinguishes Intrinsic IQ (the quality that data have in their own right), Contextual IQ (the requirement that data quality must be considered within the context of the task at hand), Representational IQ, and Accessibility IQ (both emphasizing the importance of the role of systems). Specific subcategories include

- Intrinsic IQ: Accuracy, Objectivity, Believability, Reputation
- Contextual IQ: Relevancy, Value-Added, Timeliness, Completeness, Amount of information
- Representational IQ: Interpretability, Ease of understanding, Concise representation, Consistent representation
- Accessibility IQ: Accessibility, Access security

This list was revised and extended by [5]. Here is their list:

Free-of-Error	the extent to which data is correct and reliable
Believability	the extent to which data is regarded as true and credible
Objectivity	the extent to which data is unbiased, unprejudiced, and impartial
Reputation	the extent to which data is highly regarded in terms of its source or content
Relevancy	the extent to which data is applicable and helpful for the task at hand
Value-Added	the extent to which data is beneficial and provides advantages from its use
Timeliness	the extent to which the data is sufficiently up-to-date for the task at hand
Completeness	the extent to which data is not missing and is of sufficient breath and depth for the task at hand
Appropriate Amount of Data	the extent to which the volume of data is appropriate for the task at hand.
Accessibility	the extent to which data is available, or easily and quickly retrievable
Security	the extent to which access to data is restricted appropriately to maintain its security
Ease of Manipulation	the extent to which data is easy to manipulate and apply to different tasks
Concise Representation	the extent to which data is compactly represented
Consistent Representation	the extent to which data is presented in the same format
Interpretability	the extent to which data is in appropriate languages, symbols, and units, and the definitions are clear
Understandability	the extent to which data is easily comprehended

Others add precision [2], comparability and quantitativeness [4].

Clearly, the discussions internal to the CASE team have only scratched the surface.

2.5.3 A NEW PERSPECTIVE

An important project, which we do not undertake here, is to rationalize these various schemes into a single unifying model. We suggest that a revision of our original discussion of the entities involved in the value of information may prove helpful. The basic actantial schema involves at least users, documents, hypotheses, user actions (notably decisions), and events in the external world. Figure 51 illustrates some of the relationships among these entities. Specifically,

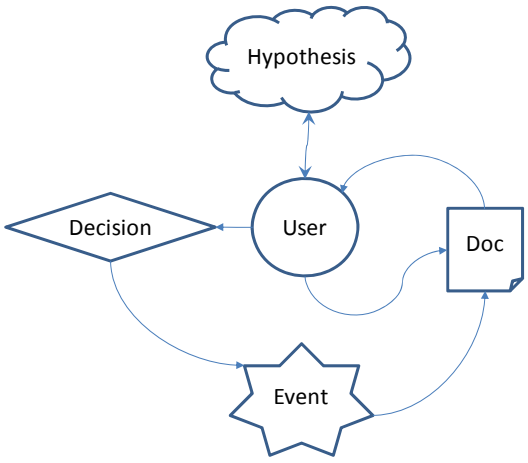

Figure 51: Entities and their Relationships

- Users both read and generate documents.
- Users formulate and consider hypotheses
- Users make decisions
- Decisions impact events in the external world.
- Events become visible to users as documents.

For clarity, the figure omits multiple copies of each of the entities, but clearly there are relations among users and other users, documents and other documents, etc.

With such a generative model in mind, we postulate that each dimension of the value of information should be defined relative to three principles:

1. We are interested in the value of *information*. Information is instantiated in our model as documents. So any aspect of the value of information can be expressed as a function whose arguments include at least one document.
2. Value of information is assessed by the *user,* which thus becomes a second essential argument to our VOI function.
3. The different kinds of value that one can associate with a document can be related to the other entities in the model. Thus it is appropriate to consider, for example, the value of one document in relation to another, or in relation to the events that led to it, or in relation to a hypothesis that is under consideration.

Such a model may help disentangle some of the dimensions that up until now have proven challenging. For example, the difference between novelty and newness may be the difference between the relation between a document and the user (novelty) vs. the relation between a document and the events that it describes (newness).

2.6 FINAL NIST EVALUATION

In September 2008 the InformANTS system supported the final NIST evaluation of the program. In the evaluation we tried to show that knowing the users' interest can augment the information retrieval quality in terms of relevance and novelty of the documents returned for a user query. In the following, we discuss the experimental setup and preliminary results.

Based on the logged data, we found evidence that a) information retrieval based on the understanding of the user has the potential to improve the outcome for the user, and b) user models are capable of revealing Virtual Interest Groups (VIG). We detail these findings in the following.

2.6.1 IMPROVED RETRIEVAL

In the final weeks of the project, we were involved in an effort to show that the InformANTS techniques have value. These experiments have concluded, and the results have shown promise. In this section, we describe the experiment design, present a preliminary summary of the data, and offer some conclusions we have drawn at this point.

2.6.2 EXPERIMENT DESIGN

We will describe briefly the experiment design and suggest motivations. While we were involved in this process, we were not the deciders, nor are we the final authority.

2.6.3 MECHANISM

With the goal of acquiring a meaningful measure of the impact of InformANTS on the experience and effectiveness of analysts' ability to do their jobs, the following basic design was used.

The user uses tools developed/refined as a part of the CASE project (nCompass) to try work on a tasking. The user attempts to learn information relevant to a tasking by searching, using the tools in the interface, and drawing/recording conclusions. As the analyst works, the system monitors the user's actions in order to build a model of the user. At set points in time during a user session, a Bin Rate Rank (BRR) form is shown to the user when the user issues a query (Adaptive Information Retrieval) or is offered recommended documents (Document Recommendation).

The BRR contains Baseline as well as Treatment documents. The user is not shown which documents originate from the Baseline set and which from Treatment. The user evaluates these documents, decided whether each is useful and, if useful, then how useful, novel, credible, and topical the document is. After evaluating the documents, the user ranks the top 10 documents (or, rather, 10 plus all equally "useful" documents, in the case of a tie on the low value end of the 10).

There are two types of BRR evaluations, Adaptive Information Retrieval (AIR) and Document Recommendation (DR).

2.6.3.1 AIR

AIR is intended to show that InformANTS can improve on the documents Google offers in response to a user's search string. The Baseline set of documents the user sees is Google's top ten documents. InformANTS attempts to process the top fifty documents and rank them. The re-ranked list of documents is the Treatment set, InformANTS' suggested documents.

2.6.3.2 DR

DR is designed to show that user models become more effective over time, as the user model grows to better reflect the user's interest. DR also depends on the query-from-user model mechanism and the re-ranking mechanism.

At the beginning of an analyst's session, a user works to build a graph representing how she is starting out on her task. Having finished this, the analyst triggers a Planning Complete message, at which point a query based on the user model at this point in time is generated and run through Google to generate a list of search results, and the re-ranked results are saved for later. When the user reaches a DR BRR, another query (from the then current user model) is generated, used to acquire Google search results, and re-ranked. The DRs come in pairs, one halfway through the user's session, and the other at the end of the user's session. Five results from re-ranking of the Google results based on the early user model query are offered as Baseline documents, and the re-ranked results from the current user model are the Treatment documents.

2.6.4 METRICS

Every session, as well as the entire experiment, is evaluated according to several metrics. We evaluate the Baseline and Treatment document sets of each BRR according to several metrics. In this report, we will focus on Value Recall, the measure of greatest interest, and the most carefully analyzed metric to date.

Keep in mind that for each document the user judges to be useful, the user assigns a "usefulness" (otherwise known as "utility") value to the document, a number between 1 and 10, where 10 indicates greatest value to the user.

Value for a system (Baseline or Treatment) is the portion of the sum of the usefulness scores assigned for which the system can claim credit. This is the sum of the usefulness for each document a system recalled, divided by the sum of the usefulness values assigned to every document in the BRR.

2.6.5 SUMMARY OF AIR RESULTS

We present only the results from the AIR sessions, because of time constraints and because analysis of the DR sessions indicated few convincing conclusions.

The experiment lasted from Thursday, September 11th, through Friday, September 19th. September 12th we assessed system performance and decided that it was necessary to change the modeler from LDA to TF, because of bad performance. (We believe that the LDA user modeling was not performing adequately, though there may have been other LDA related issues, see mutual information issues touched on elsewhere.) September 11th, and 12th were thrown out.

Additionally, a handful of other user sessions were eliminated because of BRR session timing problems, system malfunctions, and unusually small Google search result sets. Training sessions are discarded, because during training the analyst is familiarizing himself with the system, not intentionally generating a session for evaluation.

We had 33 sessions in the final set of user sessions including training sessions, but only 30 when valid user training sessions are dropped ("Accepted User Sessions" below). However, there were additional sessions where the number of documents InformANTS had to rank was insufficient. In the second row of each metrics table, we exclude user sessions where the number of documents available for re-ranking by InformANTS was low. We applied a threshold of a minimum of 21 available documents and retained 26 user sessions in the "Reranking Disadvantage Excluded" set. The threshold is arguably arbitrary, but the problems of having too few documents for re-ranking is very real.

In the following sections, we investigate Value Recall, Recall, Precision, FScore and Rank Correlation. We leave out Value Correlation because Google does not offer document value (only rank).

2.6.5.1 SUMMARY OF AIR RESULTS – VALUE RECALL

Value Recall is the percentage of total value assigned by the user that came from a particular system. For example, say the user liked two documents, giving them a value of 5 and 10. If Google suggested the document with a value of 5, but not the document with a value of 10, then Google had a value recall of .33 (33%). Program Management argued that this was the most important metric.

The simplest high level view of the AIR sessions is of "Wins." If Google had a higher Value Recall score (recalled more document value), then Google wins. Likewise, if documents suggested by InformANTS had a higher Value Recall, then InformANTS Wins.

Wins	Google	InformANTS
Accepted User Sessions	**12**	**17**
Reranking Disadvantage Excluded	**9**	**17**

Comparing the overall value recalled (sum of ALL value recalled by each system, regardless of win) is useful too:

Value Recalled	Google	InformANTS
Accepted User Sessions	**1080**	**1216**
Reranking Disadvantage Excluded	**958**	**1132**

The sum of all value recalled and the win count are useful metrics, but we sought to reinforce the observation by investigating Value Recall Win Margin. Value Recall Win Margin is the percentage of document value found by the winning system beyond the percentage recalled by the losing system. We focus on **Average** Value Recall Win Margin:

Average Value Recall Win Margin	Google	InformANTS
Accepted User Sessions	**0.210**	**0.246**
Reranking Disadvantage Excluded	**0.139**	**0.246**

This is however, not weighted by the value recalled in the document, therefore, we also look at the Average Value Recalled Win Margin.

Average Value Recalled Win Margin	Google	InformANTS
Accepted User Sessions	**9.5**	**14.706**
Reranking Disadvantage Excluded	**8.444**	**14.706**

2.6.5.2 SUMMARY OF AIR RESULTS – RECALL

Recall is the percentage of documents judged useful by the user that came from a particular system. For example, say the user liked two documents and InformANTS found one, then InformANTS has a recall of .50 (50%).

The simplest high level view of the AIR sessions is of "Wins." If Google had a higher Recall score (recalled more document value), then Google wins. Likewise, if documents suggested by InformANTS had a higher Recall, then InformANTS Wins.

Wins	Google	InformANTS
Accepted User Sessions	**9**	**14**
Reranking Disadvantage Excluded	**6**	**14**

The disparity between these scores and value recall scores are largely due to additional ties, where neither system had a greater recall.

Recall Win Margin is the percentage of documents found by the winning system beyond the percentage recalled by the losing system. We focus on **Average** Value Recall Win Margin:

Average Recall Win Margin, Weighted by Session Value	Google	InformANTS
Accepted User Sessions	**0.219**	**0.218**
Reranking Disadvantage Excluded	**0.137**	**0.218**

Again, ties have no impact. Note that, in conjunction with Value Recall, these results imply that the system recalls more valuable documents, not just more documents.

This analysis, however, is not weighted by the value recalled in the document. Therefore, we also look at the Average Value Recalled Win Margin, which results in very similar values as in the section above. The difference is that more ties are excluded.

2.6.5.3 SUMMARY OF AIR RESULTS – PRECISION

Precision is the percentage of documents suggested by the system that users deemed useful. For example, say the system suggested two documents. If the user liked one, then the system had a precision of .50 (50%).

Let's consider precision from the point of view of wins. If Google had a higher precision (the user liked more of Google's suggested documents), then Google wins.

Wins	Google	InformANTS
Accepted User Sessions	**9**	**14**
Reranking Disadvantage Excluded	**6**	**14**

Average precision margin for winners is also interesting:

Average Precision Win Margin	Google	InformANTS
Accepted User Sessions	**0.177778**	**0.214286**
Reranking Disadvantage Excluded	**0.15**	**0.214286**

Value of precision is interesting (because it seems that the sessions where the re-ranking had a small number of documents had little negative impact)… but ties are excluded again and the sample is small.

Average Precision Win Margin, Weighted by Session Value	Google	InformANTS
Accepted User Sessions	**10.26667**	**14.6**
Reranking Disadvantage Excluded	**10.35**	**14.6**

2.6.5.4 SUMMARY OF AIR RESULTS – RANK CORRELATION

Correlation is the correlation between the rank the user assigned to the documents they deemed useful and the rank the system gave to the documents. Correlation is the covariance of the ranks divided by the square root of the product of the variance of the user ranks and the variance of the system ranks.

Again, we compare wins. A system wins if it has a higher correlation with the user's ranks.

Wins	Google	InformANTS
Accepted User Sessions	**14**	**16**
Reranking Disadvantage Excluded	**12**	**15**

Further analysis here is left out because of errors in the source data and we did not have time to pin them down (though these are not at all impossible errors to deal with). The evidence seems to suggest that there is not a major difference between the performances of the two systems. Additionally, Google may even have the upper hand.

2.6.5.5 SUMMARY OF AIR RESULTS – VALUE RECALL

FScore is a commonly accepted metric for taking both recall and precision into account. We applied the following formulation, which is biased towards neither precision nor recall:

$$FScore = \frac{P * R}{2 * (P + R)}$$

As with recall and precision, InformANTS has more wins than Google:

Wins	Google	InformANTS
Accepted User Sessions	**9**	**14**
Reranking Disadvantage Excluded	**6**	**14**

Here is the comparison of win margins (although it's not clear what this means in the case of FScores). Again, InformANTS not only wins more, it wins by more:

FScore Win Margin	Google	InformANTS
Accepted User Sessions	**0.187**	**0.215**
Reranking Disadvantage Excluded	**0.143**	**0.215**

Here is what we see if we weight the FScore win by the value of the session. Again, the meaning of relative FScores in this context is unclear:

Average FScore Win Margin, Weighted by Value of Session	Google	InformANTS
Accepted User Sessions	**10.12**	**14.54**
Reranking Disadvantage Excluded	**9.80**	**14.54**

2.6.6 EVIDENCE SUGGESTING STRONGER AIR RESULTS

Each user session during the evaluation experiments run by NIST had two AIR BRR evaluations. The first, 'AIR1', was done early after the user had completed planning based on their tasking and had used nCompass to begin to seek the answers to the tasking. The second AIR, 'AIR2', was done after additional research by the analyst, at the end of the session and immediately before the second and final DR. InformANTS performs very well during the AIR1 BRRs. However, Google edges out InformANTs in the AIR2 experiments.

Wins	Google	InformANTS
AIR1 BRRs	**2**	**12**
AIR2 BRRs	**9**	**7**

This marked difference is either random or it has a specific cause. If it is random, it sheds doubt on the credibility of the experimental results. Although the results are strong, we have no way of saying how confident we are that the results accurately represent the value of the algorithms or the performance of the system.

On the other hand, if there is a reason for the difference in system performance among the AIR1 and AIR2 BRRs, then real potential of the system performance is actually very likely the AIR1 performance. And, moreover, if we could modify the algorithms or system appropriately with this knowledge, the system performance might always be that strong or easily be far better.

Therefore, we worked to understand what the cause might be. We present here two reasonable hypotheses. Each strongly correlates to the evidence. And although we have no proof of causality, given additional time we could get closer to proof.

2.6.6.1 HYPOTHESIS 1 – USER MODEL CHANGES

A user model could change consistently between AIR1 and AIR2, because it is built on additional ALEs as the user works in-between DR1 and AIR1. Therefore, we sought to find differences between early and late models and ask how differences could affect system performance.

We reconstructed the user models from the user model tracing done during the experiments. When the system runs, a new user model is recorded every time there is a user model update.

We analyzed the user models to look for consistent patterns of change during the course of the experiments. What we found was that entropy and model size both change consistently during the course of the experiment. Additionally, this could very well account for diminished system performance for AIR2, with respect to AIR1.

Model size grew steadily from a handful to well over 1000 markers for some models. The chart in Figure 52 is a visualization of this, for a set of 20 models from the final four days of the NIST-run experiment.

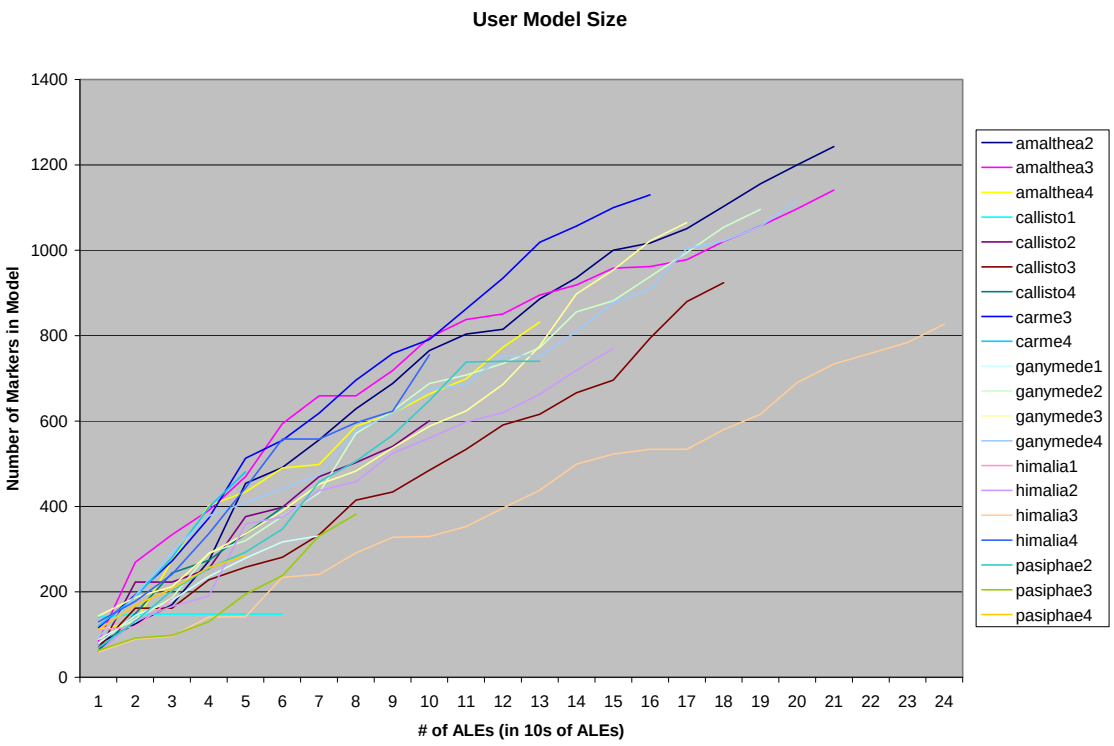

Figure 52. User Model Size over a Series of ALE

Note that this chart has an x-axis of ALEs, not time. Every user session was about the same length. It is fair to say that most users grow their model at roughly a rate of growth. (It is interesting to note, from this chart, that the user models tended to grow at roughly constant rates, with respect to number of ALEs. Consider the short (in ALEs) models that tend to have steeper slope; one could guess that these are users who tended to get longer documents or read instead of search. Therefore, they read fewer documents, but had on average a greater number of new markers per ALE. This type of analysis could be useful in the future.)

These user model sizes are very large. With the modeler in question, the TF modeler, object models from documents are an order of magnitude smaller. User models do not need to be of the same size as document models (this is not topic modeling, where every user and document is a distribution over the same topics), however, this large size *does* matter.

Entropy decreased steadily from random (our interest modeling assigns to the first model the same weight for every term, which, when normalized to be a probability distribution, is a random distribution). The chart in Figure 53 is a visualization of this steady decrease, for a set of 20 models.

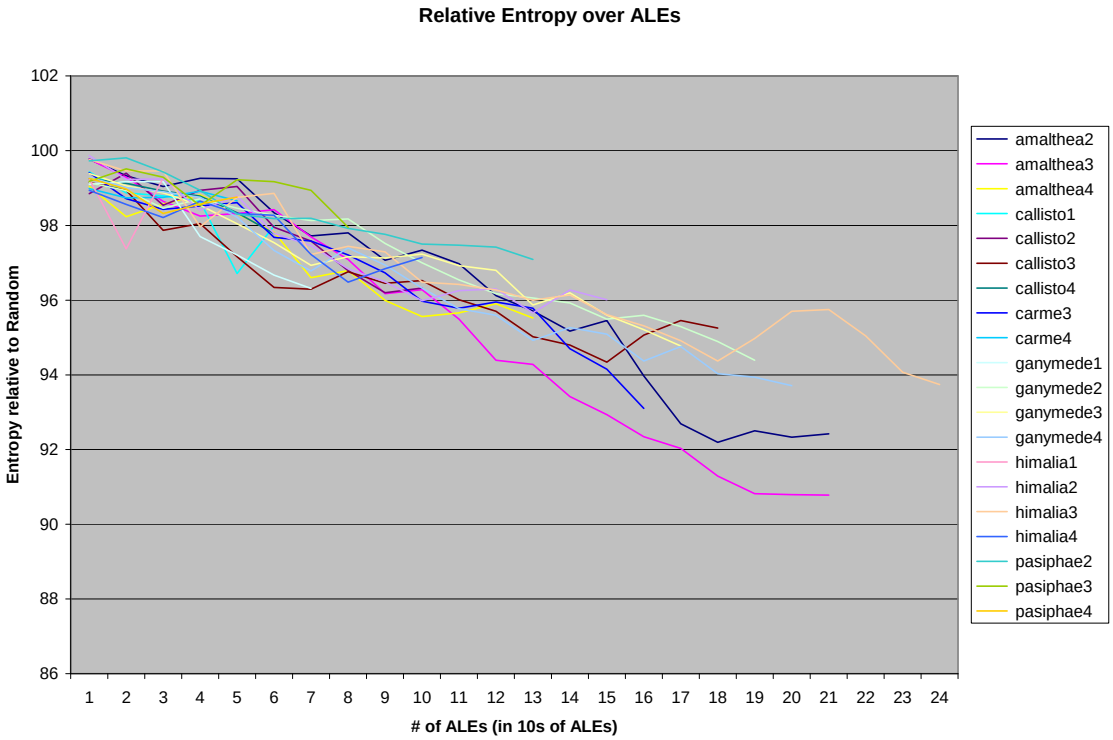

Figure 53. Normalized User Model Entropy over a Series of ALE

Again, the chart has an x-axis of ALEs, not time. And, again, all user sessions were of roughly the same length, in terms of time.

We calculated entropy with respect to the entropy of a random distribution of the same size, in number of terms, as the user model:

$$H_{w.r.t.random}(X) = \frac{-\sum_{i=1}^{n} p(x_i) \cdot \log(p(x_i))}{-\log\left(\frac{1}{n}\right)}$$

The numerator is the entropy of the distribution; the denominator is the maximum entropy of a uniform distribution over the same number of terms. This normalization makes the measure independent of the number of samples as well as the base of the logarithm. We look at entropy with respect to random because it lets us see how "close" models of different sizes are to random. This is one way to make models of very different sizes somewhat comparable.

Given the steady growth over time to quite large user models, and the consistent lowering of entropy as time progresses and the models grow, do we have reason to believe that these user model variations suggest a cause for the AIR1-AIR2 differences? Consider how user models are used in the document ranking algorithm. Every time there is a user model update, InformANTS finds the subset of documents most near to the user model in the matching space and applies forces derived from similarity between those documents and the user model. Because the user model is broad and relatively close to random, it is possible that this selection of documents is less effective because the large flat model is no longer useful for the similarity calculations or the forces on the selected documents are muddled, again, because the model is too large/flat. *However*, it is also possible that a large user model of this entropy is not a problem at all, we have no proof to the contrary; knowledge of how the ranking is done *does not automatically lead us to the conclusion that large models of this entropy negatively impact the effectiveness of the re-ranking*. The hypothesis remains merely a hypothesis.

We do have good reason to believe that if the user model size/entropy were the cause of the problem, we could certainly achieve AIR1 quality performance in general. Control by user modeling parameters (e.g. aging), size of model, or entropy (e.g. warp the distribution to exaggerate/dampen the distribution to a desired entropy) would likely yield models that would allow the system to consistently perform as well as the system did during AIR1. Moreover, we are fairly confident that we could achieve AIR1 quality results if the user modeling hypothesis is true because we could always use exactly the same algorithm as was used during the NIST Evaluation, yet only present a model built from the number of ALEs typically generated before AIR1 (forget older ALEs).

2.6.6.2 HYPOTHESIS 2 – DOCUMENT RERANKING

Another difference between the state of the system when the AIR1 and AIR2 BRRs are done is the number of documents in the matching space.

Because of timing and budget constraints, we could not pursue evaluation of this hypothesis. Based on the evidence we have, it is feasible in much the same way as the first hypothesis: there *is* a correlation, but we do not have proof of causation. Likewise, we can also easily imagine how the number of documents could hurt system performance, but do not have evidence that what we imagine could have happened, happened.

To pin this document cluttering as a cause, we could calculate from the saved experiment data how many documents were in the Locator at any point in time. We would expect to see a correlation to number of users using the system and the severity of the difference between AIR1 and AIR2 performance. Additionally, we could look deeper, at specific metrics of system performance. And, of course, the ability to run the system at will with live users to evaluate hypothesis and solutions would be ideal.

If we knew that documents were cluttering the system and hurting performance, we could work on the organization techniques. Again, we are fairly confident we could deal with the performance even if a solution for more documents was not feasible: we could always do re-ranking without older documents in the information space.

2.6.6.3 SUMMARY

There is a startling disparity between AIR1 and AIR2 performance. When we investigated possible causes, we came up with two very feasible hypotheses. Unfortunately, we don't have proof of causation. However, if either of these hypotheses are the real cause, we believe we could achieve AIR1 levels of performance in general.

2.6.7 VIG IDENTIFICATION

As a note of caution, at the time of this writing, we do not have knowledge of the ground truth. In other words, we don't know the correctness of the task identification (aka VIG identification). This section is based on data that we had.

Now we present our task identification results in two steps. First, we describe the findings when we focus only on the 28 users participated the final evaluation experiment. We were able to clearly identify 4 tasks that the 28 users seemed to work on by the user models built with the TF and the T2SCM modelers. The tasks made by the other modelers were not as clear or well-defined. Second, we show the results with the current 28 users as well as the 34 historical users. Here we see 6 tasks including the 4 we saw before. Again the TF and T2SCM seemed to show well-defined tasks whereas the other models did not.

2.6.7.1 RESULTS USING USERS FROM THE FINAL EVALUATION EXPERIMENT

Here we describe the findings when we focus only on the 28 users who participated the final evaluation experiment.

2.6.7.1.1 USERS AND THEIR MODELS

These users are:

io1-4; himalia1-4; ganymede1-4; pasiphae1-4; callisto1-4; carme1-4; amalthea1-4

We built up to 8 half-hour models built per user. These models were built as the 30-min segment models. As a result, we had a total of 168 segment models for task identification. For example, user amalthea1 has 6 segment models built:

```
amalthea1.30: using ALEs during time 0-30 min
amalthea1.60: using ALEs during time 30-60 min
amalthea1.90: using ALEs during time 60-90 min
amalthea1.120: using ALEs during time 90-120 min
amalthea1.150: using ALEs during time 120-150 min
amalthea1.180: using ALEs during time 150-180 min
```

2.6.7.1.2 VIG MEMBERSHIP MAP (VMM)

The first step of task identification is to find the VIG for each segment model. For segment model amalthea1.90 and a VIG size of 5 we found the VIG members are:

```
amalthea1.90:
1. {io1.150, 0.18573456}
2. {amalthea1.30, 0.15227921}
3. {carme1.180, 0.13571176}
4. {amalthea1.180, 0.12650149}
5. {himalia3.60, 0.12279697}
```

In particular, io1's segment model io1.150 is the most similar model with similarity value of 0.186 whereas user himalia3's segment model himalia3.60 is the least similar of the 5 members with a similarity value of 0.123. Note the similarity value ranges between 0 and 1, where larger values indicate stronger similarity of the concerned models.

Next step involves combining the VIG members of the segment models generated from the same user. Here we count the number of VIG members contributed by each user. In other words, we find out the VIG membership profile for each user. For example, after combining the VIGs of amltheal1's segment models, we got the following VIG membership profile: 8 members from user io1, 1 from himalia1, 3 from carme1, 17 from herself, and 1 from himalia3. The other 24 users did not contribute any segment models to the amltheal1's VIGs.

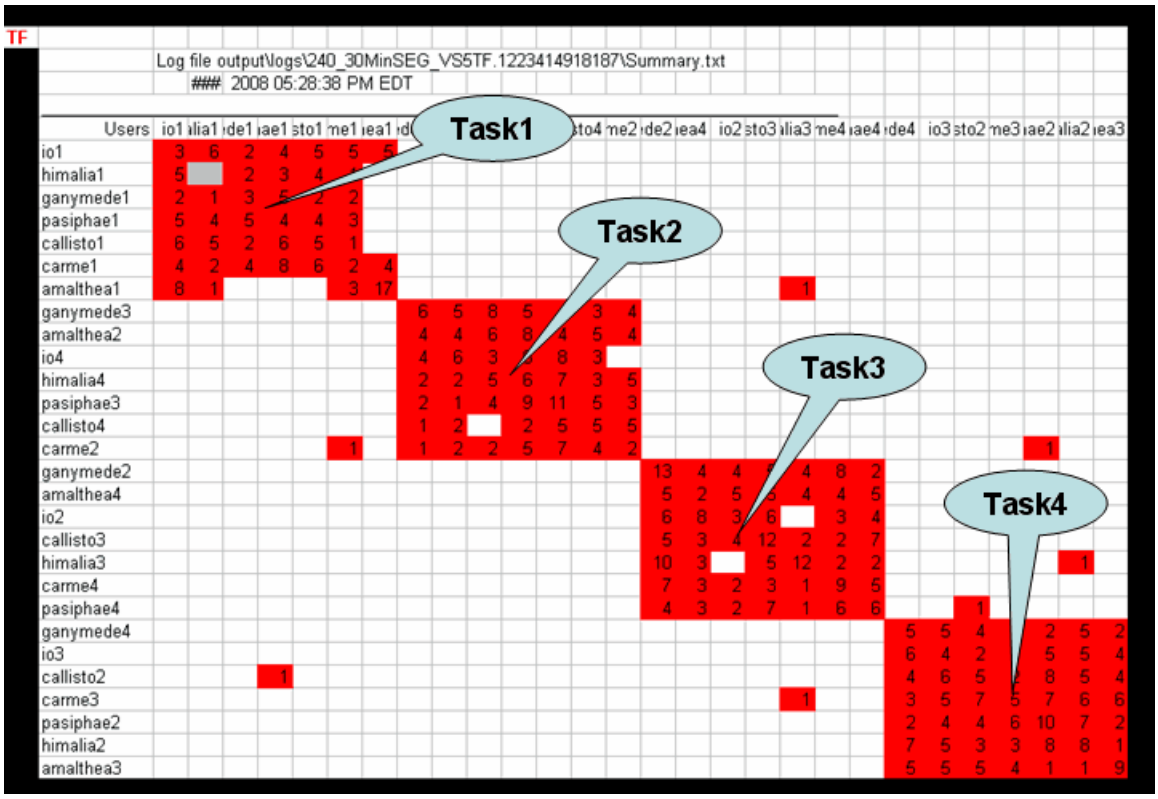

Figure 54. Task identification assessed at 240 min using 30-min segment models with TF modeler.

The VIG membership profiles are then put into a VIG membership map (VMM) (Figure 54). In a VMM, each row represents the VIG membership profile of the user identified by the row title. The value of each cell of that row represents the number of VIG members that come from segment models of the user identified by the corresponding column title. If we examine the row for user amltheal1, we see that first cell has a value of 8 and corresponds to the io1 column. That means io1's 8 segment models have ended up in the VIGs of amltheal1's segment models. Note that for better visual effects, the cells with values equal to or greater than 1 are shaded bright red in a VMM.

By rearranging the VMM such that the rows with similar VIG membership profiles (i.e. rows) are next to each other, we expose the user clusters where each cluster identifies an underlining task that these users are working on. In Figure 54, the VMM of the 28 NIST evaluation users are shown and four user clusters or tasks are clearly identifiable.

2.6.7.1.3 TASK IDENTIFICATION

In Figure 54, the VMM of the 28 NIST evaluation users are shown. The 30-min segment models were built using TF modeler. The VMM shows the results at time 240 min, i.e. at the end of the experimental sessions. In the VMM, four user clusters or tasks are clearly identifiable. They are described in detail below.

2.6.7.1.3.1 TASK1

io1, himalia1, ganymede1, pasiphae1, callisto1, carme1, amalthea1

2.6.7.1.3.2 TASK2

ganymede3, amalthea2, io4, himalia4, pasiphae3, callisto4, carme2

2.6.7.1.3.3 TASK 3

ganymede2, amalthea4, io2, callisto3, himalia3, carme4, pasiphae4

(amalthea1) [9]

2.6.7.1.3.4 TASK 4

ganymede4, io3, callisto2, carme3, pasiphae2, himalia2, amalthea3

2.6.7.1.4 TASK IDENTIFICATION OVER TIME WITH TF MODELER

[9] The bracket indicating the user's association with the task is secondary. In other words, its primary task is elsewhere.

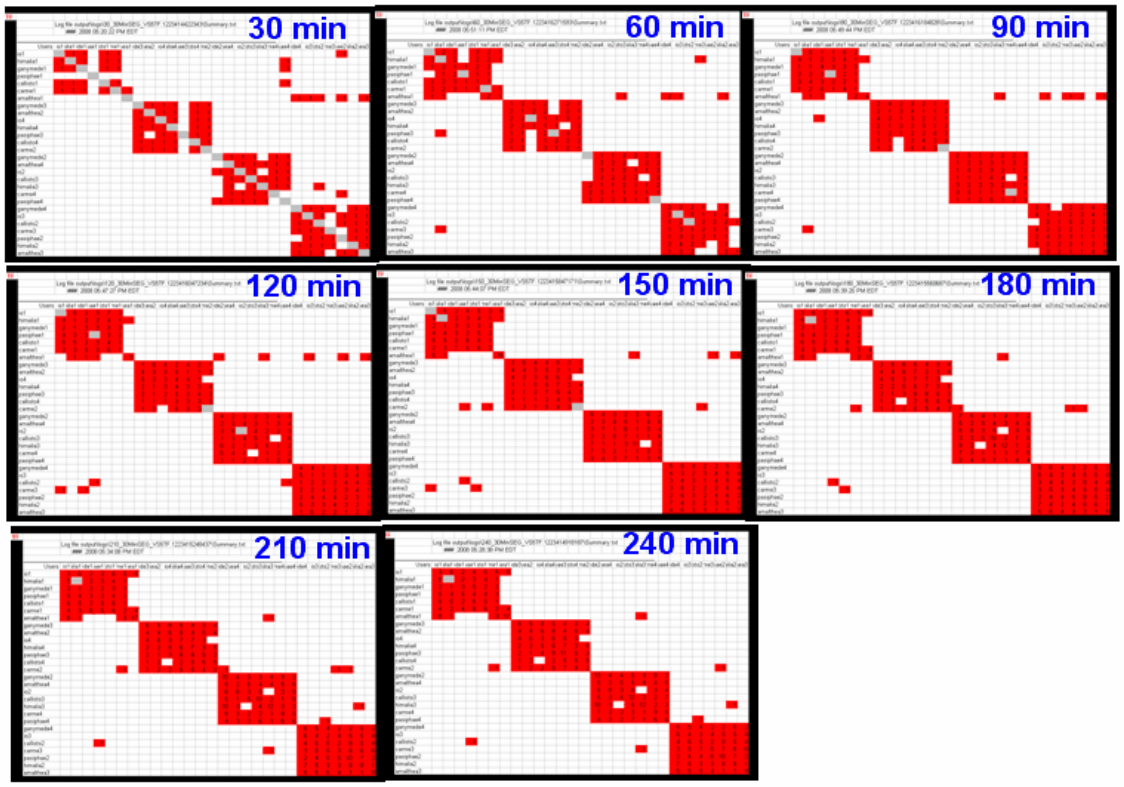

Figure 55. The effect of time on the task identification with TF modeler.

The VMMs with TF modeler over time at 30-min intervals are shown in Figure 55. The clusters seem to look tighter over time. In other words, the task identification seems to get better over time.

2.6.7.1.5 VIGS OVER TIME WITH T2SCM

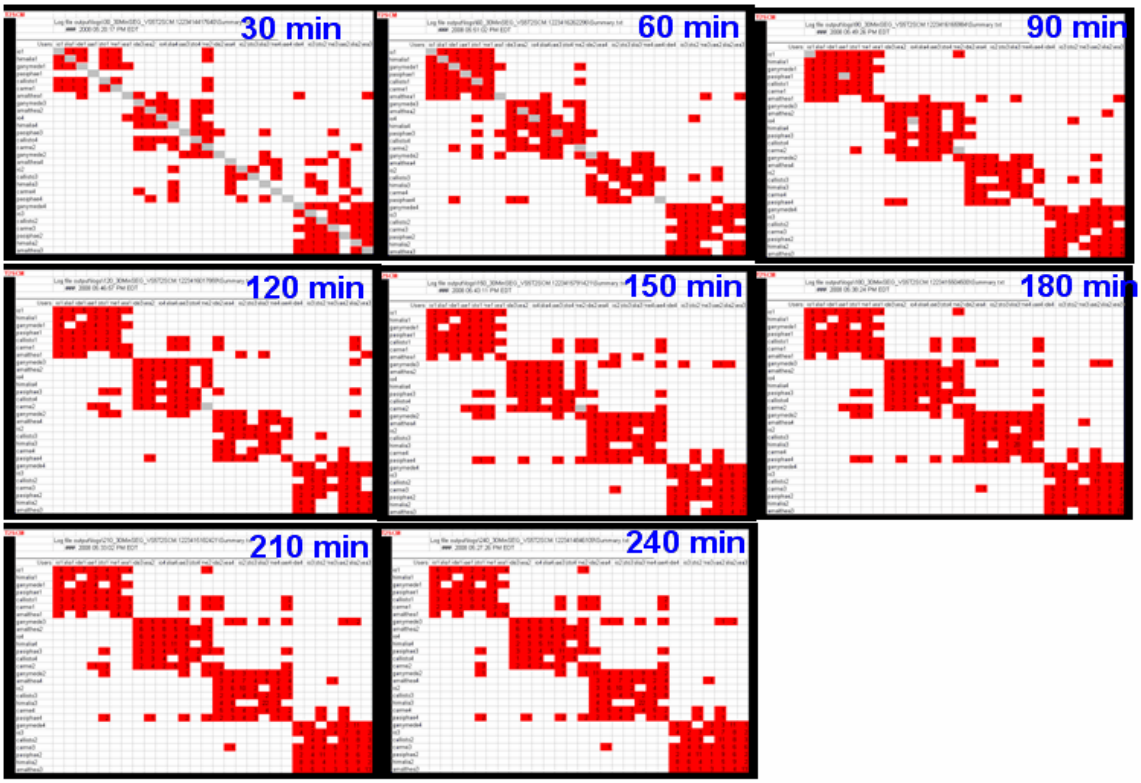

Figure 56. The effect of time on the task identification with T2SCM modeler.

The VMMs with T2SCM modeler over time at 30-min intervals are shown in Figure 56. Even though the overall results are not as clean as we saw with TF modeler above. The clusters here again seem to look tighter over time. That is to say that the task identification seems to get better over time.

2.6.7.1.6 VIGS BY 5 MODELERS AT TIME 240 MIN

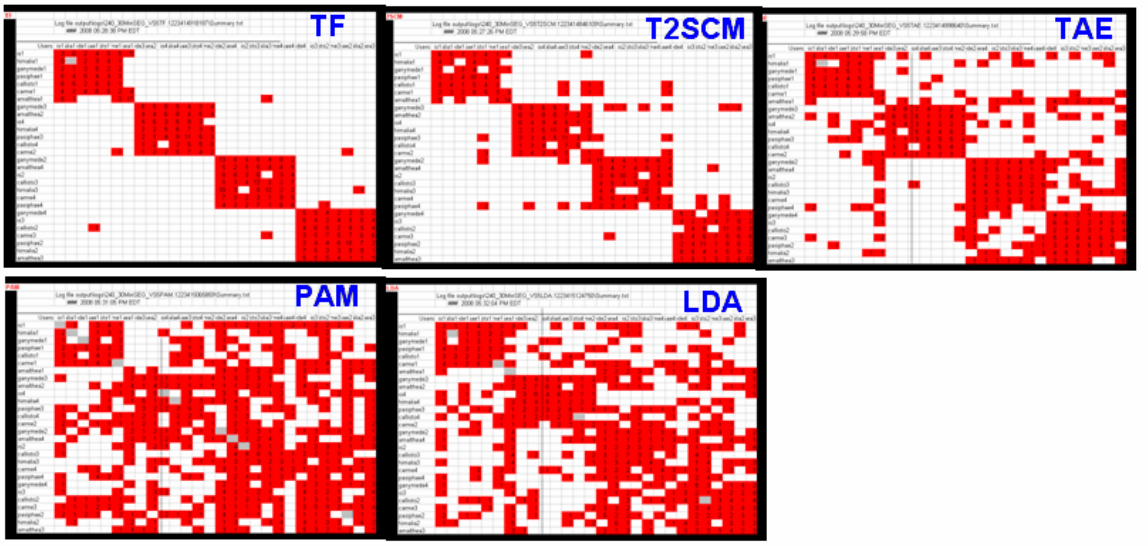

Figure 57. The effect of modelers on the task identification at time 240 min.

The VMMs with all 5 object modelers at time 240-min are shown in Figure 57. We can clearly identify 4 tasks that the 28 users seemed to work on by the user models built with the TF and the T2SCM modelers. With TAE, we can still make out the tasks. However, the tasks made by the PAM and LDA modelers were not as clear or well-defined.

2.6.7.2 RESULTS USING THE FINAL EVALUATION EXPERIMENT USERS AND HISTORICAL USERS

Now we describe the findings using both the 28 evaluation experiment users and the 34 historical users.

2.6.7.2.1 USERS AND THEIR MODELS

For information on the users and models for the 28 evaluation experiment users, please see Section 2.6.7.1.

The historical users include:

- Glass Box Users: ccgbuser4,7; lc1-6; ltrist1-5
- APEX Users: APEXB, APEXC, APEXE, APEXF, APEXH, APEXK, APEXP, APEXD, apexd, APEXL, APEXN, APEXG, APEXI, APEXJ, APEXO
- Test Users: elara1-4; thebe1-2

For historical users with multiple days (Glass Box users and APEX users), two segment models are built for each day per user. The first model is built with the first 50 ALEs and the second with the next 50 ALEs.

For historical users with only one day (Test Users), only two segment models are built for each user with the first and second 50 ALEs. We have built 225 segment models for historical users and 168 for the evaluation experiment users. Together, we have a **grand total of 393 segment models for analysis.**

2.6.7.2.2 TASK IDENTIFICATION

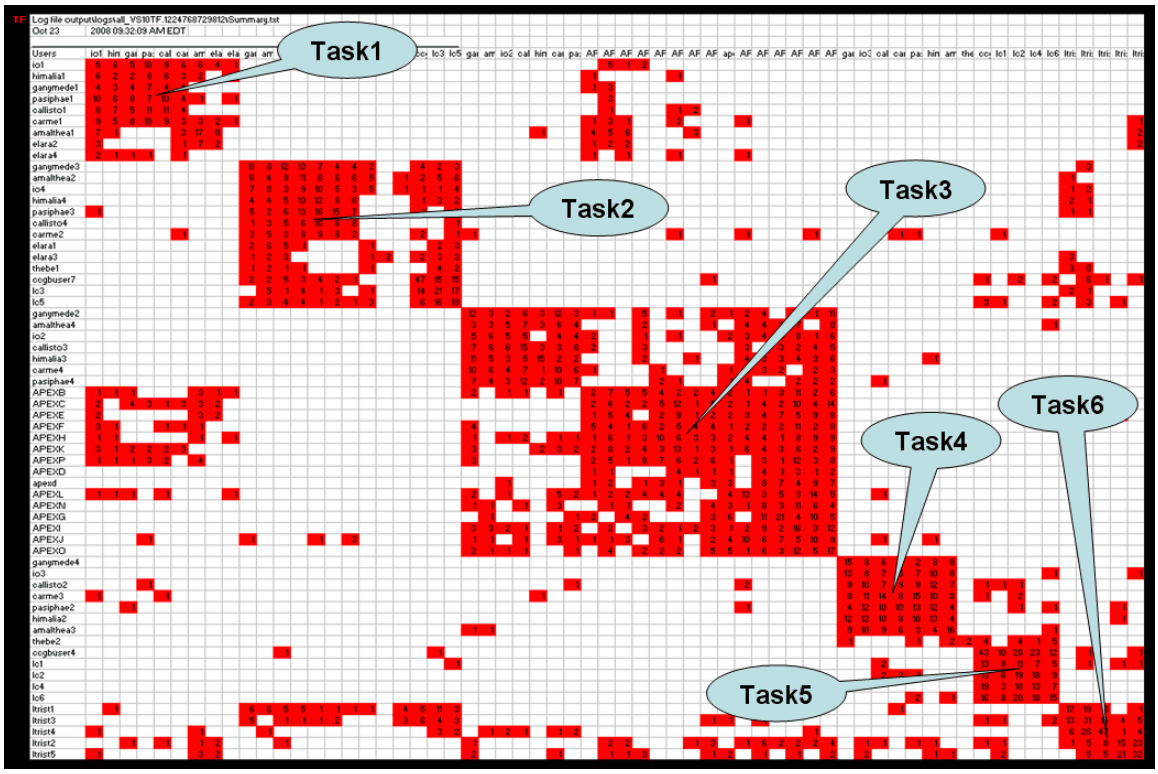

Figure 58. The Task identification for all 62 users from current and historical experiments.

In Figure 58, the VMM of all 62 users with TF modeler are shown. For the evaluation experiment users, the VMM shows the VIG membership profiles at time 240 min, i.e. at the end of the experimental sessions. In the VMM, six user clusters or tasks are clearly identifiable. They are described in detail below.

2.6.7.2.2.1 TASK1

io1, himalia1, ganymede1, pasiphae1, callisto1, carme1, amalthea1, elara2, elara4

(APEXB, APEXC, APEXE, APEXF, APEXH, APEXK, APEXP, APEXD, apexd, APEXL)[9]

2.6.7.2.2.2 TASK2

ganymede3, amalthea2, io4, himalia4, pasiphae3, callisto4, carme2, , elara1, elara3, thebe1, ccgbuser7, lc3, lc5

(ltrist1, ltrist3, APEXJ)

2.6.7.2.2.3 TASK 3

ganymede2, amalthea4, io2, callisto3, himalia3, carme4, pasiphae4

APEXB, APEXC, APEXE, APEXF, APEXH, APEXK, APEXP, APEXD, apexd, APEXL, APEXN, APEXG, APEXI, APEXJ, APEXO

(ltrist4, ltrist2, ltrist5, amalthea1)

2.6.7.2.2.4 TASK 4

ganymede4, io3, callisto2, carme3, pasiphae2, himalia2, amalthea3

(thebe2)

2.6.7.2.2.5 TASK 5

thebe2, ccgbuser4, lc1, lc2, lc4, lc6

2.6.7.2.2.6 TASK 6

ltrist1, ltrist3, ltrist4, ltrist2, ltrist5

2.6.7.2.3 EFFECT OF TIME

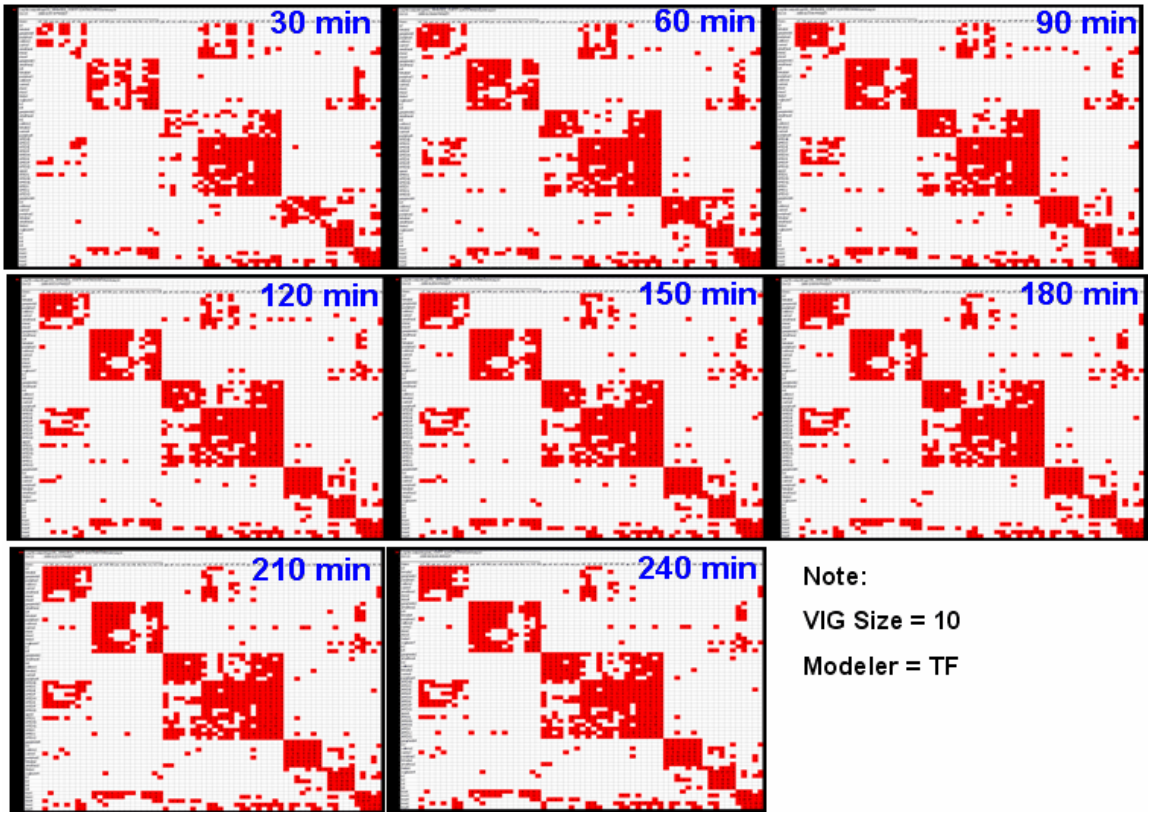

Figure 59. The effect of time on the task identification.

The VMMs with TF modeler over time at 30-min intervals (for the evaluation experiment users) are shown in Figure 59. The task identification seem to get better over time from time 30 min to 120 min. Beyond time 120 min, it's hard to discern any improvements.

2.6.7.2.4 EFFECT OF MODELERS

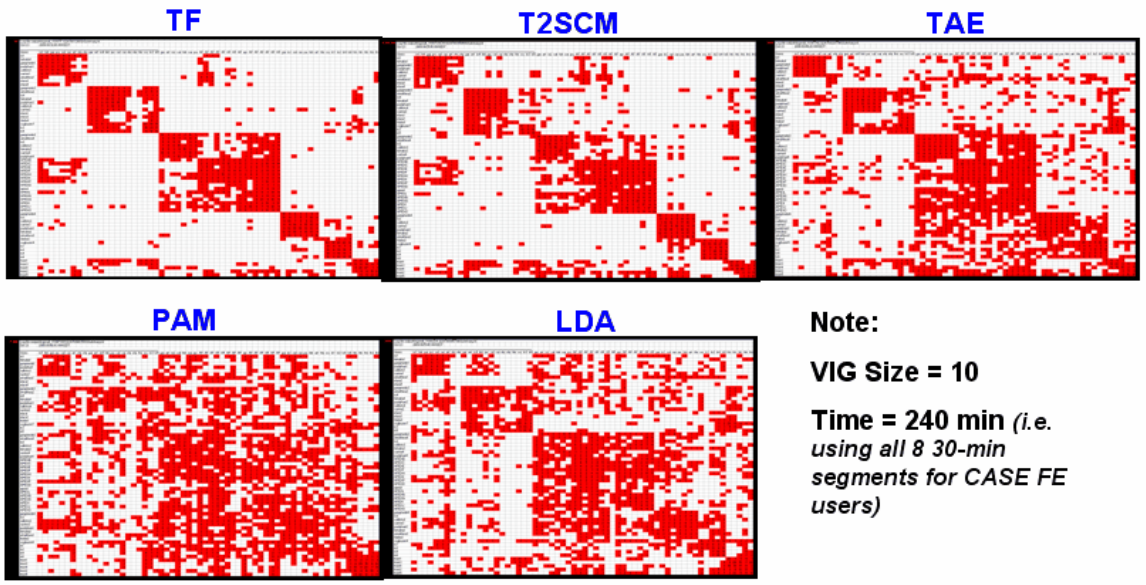

Figure 60. The effect of modelers on the task identification.

The VMMs with all 5 object modelers at time 240-min (for the evaluation experiment users) are shown in Figure 60. We can clearly identify 6 tasks that the 62 users seemed to work on by the user models built with the TF modeler. With T2SCM modeler, the 6 tasks can still be identified but the boundaries are less clear. With TAE, we can still make out the tasks, though not as good as with T2SCM. The tasks made by the PAM and LDA modelers were not as clear or well-defined.

2.6.7.2.5 EFFECT OF VIG SIZE

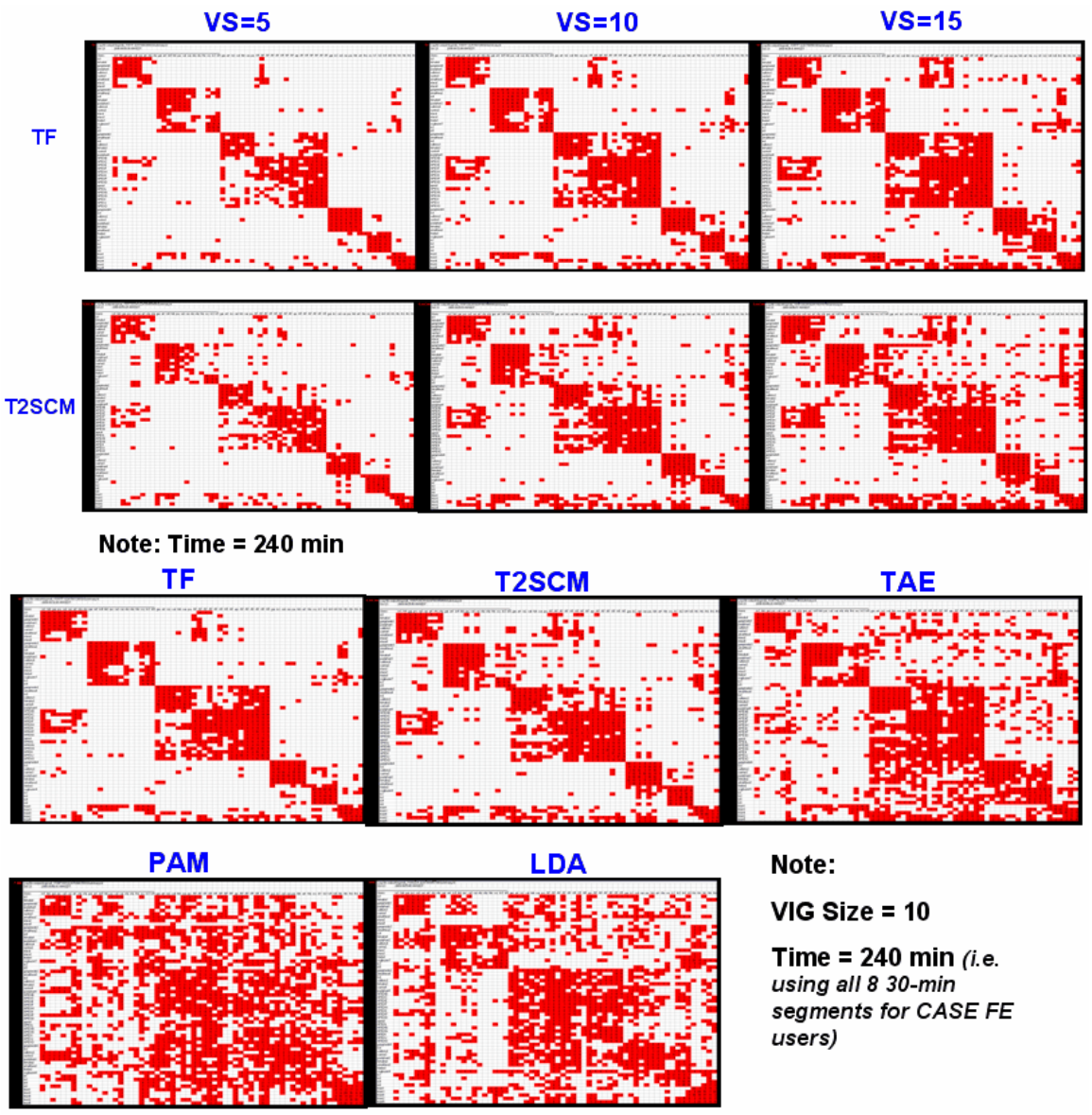

Figure 61. The effect of segment model VIG size on the task identification for TF and T2SCM modelers.

Figure 61 shows the effect of the VIG size used when building the VIG for each segment model in the process of creating the VMM. A matrix of VMM for three VIG size (5, 10, and 15) by two modeler (TF and T2SCM) is shown. With TF modeler, VIG size of 15 seems to give the best task definition. With T2SCM modeler, VIG size of 10 and 15 are about similar and both are better than VIG size of 5.

3 LESSONS LEARNED

Overall, we assert that our project has exceeded our expectations in regards to the goals that we had set for the first two years of the initially proposed four-year effort. We learned that to achieve these goals, we had to maintain a rigorous focus in our software engineering component of the project and subjugate the scientific progress to the needs of specific milestones and the availability of data and data-preparation tools.

In the following, we touch on some of these issues in more detail.

3.1 SCIENCE VS. ENGINEERING

Any R&D project that goes beyond "pure" (theoretical) science and tries to have a real-world impact must balance the need for scientific progress with the requirement to capture the output of the research in functional prototypes that can be demonstrated and whose value can be evaluated. Naturally, this balancing act applied to the CASE MASTER project too, where we had to make scientific progress on several fronts (e.g., dynamic user modeling, self-organizing data) while demonstrating the impact of this progress on increasingly complex prototypes that are integrated with products from other CASE teams.

We learned from experience in previous related projects that focusing on selected scientific problems and simplistic stand-alone demonstration prototypes first runs the risk of a) not being relevant to the needs of the program as a whole, and b) not being prepared for the problems that dealing with real-world data adds to the picture. Therefore, in CASE MASTER we put engineering before science in the sense that we first defined the capabilities that a next prototype would bring to the table in the CASE integrated system. Only then would we identify the "gaps" in our current knowledge that would have to be filled by the scientific endeavor to meet these capability requirements. Sometimes, these constraints came in conflict with our desires to follow interesting research leads, but the approach eventually enabled us to leave the curtailed project with solid evidence that the InformANTS concept as a whole works as proposed.

3.2 SELECTING THE APPROPRIATE INFRASTRUCTURE

The project produced a series of prototypes with increasing complexity, scalability, and ability to handle real-world data. Individual components, such as the User Modeling System or the Information Matching System, initially evolved rather independently with only loose couplings in their interactions, but as their capabilities increased, tighter coupling offered significant synergies. Therefore, different phases of the project had different requirements on the internal integration infrastructure. Early on, the integration requirements were small with infrequent and low-volume interactions, but in the final prototype, InformANTS-internal traffic increased by order(s) of magnitude.

Thus, it is an important lesson from this project to remain aware of the integration requirements of the evolving prototype and to be prepared to change the underlying communications, execution, and data infrastructure if needed. Specifically, in CASE MASTER, we moved from the execution of our main components as independent webservices to a tight integration of more robust Java processes through an xmlBlaster infrastructure.

3.3 IMPORTANCE OF DATA

The tendency of science-oriented projects is to produce abstract, stand-alone, and small-scale demonstrations that highlight particular achievements or confirm theories. Naturally, these demonstrations work best in a tightly scripted environment with carefully selected and often artificially created data. But it is exactly this tight control of the scenario that prevents such projects from demonstrating real-world relevance because all the problems that come with noisy, large-scale, and dynamic data and environments are "ignored away".

In this and other projects, we learned to "get our feet wet" and work with realistic scenarios early on as a means to guide and prioritize our scientific endeavor. Again, just as in the general balance between science and engineering (section 3.1), the data that is actually available (rather than just postulated) and any tools that are required to pre-process that data determine what scientific problems need to be solved to achieve a desired set of demonstrable capabilities.

A case in point is our proposed use of Specialized Concept Maps as the basic foundation for user and document models in the InformANTS prototype. The reality of the available data and tools forced us to abandon (for now) our idea a) to have specialized relations between specialized concepts (e.g., Person A "is located at" Place B), and b) to associate our specialized concepts with a rich context of entity data. Instead, we reduced relations between concepts to co-occurrence in text because sufficiently sophisticated entity-extraction tools were not available. We also limited the concept characteristics (and subsequently the reasoning over these characteristics) to a few numerical attributes such as longitude/latitude for Place concepts or age/gender for Person concepts, because we had access to public databases for such broad characteristics.

4 DIRECTIONS FOR FUTURE RESEARCH

In the following, we discuss a few topics that could warrant further research and development.

4.1 AUTOMATIC QUERY GENERATION FROM USER MODEL

Earlier we have mentioned that this issue is complex and deserves research in its own right. There are many strategies by which to generate queries from a given user model. In a preliminary study, besides from the "engineering hack" adopted in the NIST evaluation, we have looked into 5 alternative queries generation strategies as alternatives and made comparisons for their effectiveness. In this study, there are 6 top markers from the user saniya33's model, each with 3 terms. The markers are listed below in descending order of weight:

1. chemical biological weapons
2. united world international
3. russia russian soviet
4. missile nuclear missiles
5. nuclear north iaea
6. smallpox anthrax biological

The original and 5 alternative queries are described below.

Q1. The original query.
[chemical united russia missile nuclear smallpox]
Q2. AND query of all terms excluding duplicates (Note By default, Google only returns pages that include all of your search terms. There is no need to include "and" between terms.)
[chemical biological weapons united world international russia russian soviet missile nuclear missiles north iaea smallpox anthrax]
Q3. OR query of all terms excluding duplicates.
[chemical OR biological OR weapons OR united OR world OR international OR russia OR russian OR soviet OR missile OR nuclear OR missiles OR north OR iaea OR smallpox OR anthrax]
Q4. OR terms within a marker, then AND all markers.
[(chemical OR biological OR weapons) (united OR world OR international) (russia OR russian OR soviet) (missile OR nuclear OR missiles) (nuclear OR north OR iaea) (smallpox OR anthrax OR biological)]
Q5. Pick number of terms according to weights. All 3 terms of the top 1/3 makers, first 2 terms of the mid 1/3 markers, and the first term of the bottom 1/3 markers are picked and AND-ed together.
[chemical biological weapons united world international russia russian missile nuclear north smallpox]
Q6. Pick number of terms according to weights variation. All 3 terms of the top half makers, and the first term of the bottom half markers are picked and AND-ed together.
[chemical biological weapons united world international russia russian soviet missile nuclear smallpox]

Emile Morse of NIST was kind enough to run a Google search with these queries and looked through the query results. She rated the performance of the queries as follows:

Q1. B+
Q2. F
Q3. D
Q4. C
Q5. D
Q6. D

In addition, in the case of Q1, Emile thought, nearly all the documents would have been useful. Based on Emile's judgment with these queries, the original query (Q1) is doing as well as if not better than the alternative queries. Emile also cautioned about her ratings. She said, "There is a huge issue in that I read the results of query 1 first and then went step-wise through to query 6. By that time, I'd seen most of the documents somewhere. It is extremely hard to hold a constant scale under these conditions."

The preliminary study is by no means conclusive. We only had one query from each strategy and the results can hardly be generalized. Further research and experiments are required to find the best yet elusive query generation strategy.

4.2 DOCUMENT RELEVANCE AND NOVELTY

We have begun to research the issue of relevance vs. novelty for a document from the perspective of user modeling. One insight is that relevance is tied to user's interests on topics whereas novelty to user's knowledge on the topics. Thus a document is relevant if it touches upon topics the user cares about. A document becomes novel if it discusses either new topics previously unknown to the user, or unfamiliar topics of which the user is aware but have little knowledge. Another insight is that the user models ought to capture both interests and knowledge in order to access both relevance and novelty of a document. Our current user model captures user interests but not user knowledge. One possible extension is to add a familiarity weight to the topics in the user model to indicate how much knowledge the user has for the particular topic. A third insight is that given a user model capturing both interests and knowledge, a document can be assessed for the information value of relevance and novelty automatically.

4.3 FRONT PAGE NEWS PREDICTION

This problem is an interesting application of user modeling. The idea is about predicting front page news for a particular publication such as New York Times, which is available through the Factiva data set distributed by the CASE management. We assume that the front page editors have a stable set of values governing what story or news should go to the front page. Given the Factiva data set, we know which articles are printed on which page. We can create two subsets of data: a training set with the front page news known, and a testing set with the articles' page information withheld. The evaluation is based on the precision and recall of on the prediction whether a test article is front page news.

4.4 DYNAMICS OF SELF-ORGANIZING INFOPACKS

For the first time in the NIST-evaluation prototype, we applied the force-based self-organizing arrangement of InfoPacks to sufficiently large sets of agents (InfoPacks) with a rich set of forces. While we were able to manually tune the system to provide a reasonable arrangement of documents relative to a user to perform the re-ranking function, the overall matching of documents to documents to users to users (full two-by-two matrix) that emerged in this process was less than satisfactory and not yet sufficient to reach the full potential of the InformANTS vision.

The reason for this shortfall is in the complexity of the dynamics of the self-organizing process, which, as we found in the tuning phase before the evaluation, has a strong dependency on the topology of the graph spanned by the forces among the InfoPacks. We believe that further research is necessary to understand this dependency and to develop local adaptation techniques applied to the individual agents in the process that would control the effective topology of the graph for optimal matching performance. In particular topology parameters such as the minimum embedding dimensions, minimum longest path, and individual node dimensionality appear to be most influential.

4.5 WEB2.0 COLLABORATION AS A STIGMERGIC PROCESS

One of the deployment visions for InformANTS is the enhancement of Web 2.0 collaboration in specialized Wiki's that may not have the critical mass of active users and contributors to establish the feedback dynamics that lead to the emergence of vibrant and rich content as we see it for instance in Wikipedia. Successful wikis rely on the user community's ability to rapidly and reliably identify missing, out-of-date, or incorrect content across a large number of pages (Wikipedia currently claims 2.5 million articles in English alone), to identify the appropriate contributors to enhance the content, and to collaborate across virtual interest groups in the expansion of the knowledge in the repository. In other words, the community needs to self-organize to match users and contributors with documents. These self-organizing processes among human users that are geographically dispersed and that otherwise do not interact with each other need to be shaped appropriately through rules and tool/interface support to achieve the desired emergent dynamics (maintaining vibrant and rich content) for the community.

Self-organizing human user populations in Web 2.0 environments can be modeled as stigmergic processes [11] where autonomous actors interact indirectly through a shared environment to achieve (consciously or not) a common goal. Mapping the specific dynamics of Web 2.0 collaboration into such a formal model and then analyzing its performance characteristics stand-alone and as impacted by collaboration tools such as InformANTS will provide qualitatively new insights into the design of special-purpose collaborative communities with immediate impact on the Intelligence Community.

An additional issue surrounding facilitated collaboration is the possibility of collective cognitive convergence discussed above in the context of VIGOR. Tools to monitor, measure, and where necessary manage this convergence are essential to ensure that the benefits of Web 2.0-enabled stigmergy are not lost to cognitive collapse.

5 TECHNOLOGY TRANSITION STATUS AND ACCOMPLISHMENTS

In the curtailed (two-year) CASE MASTER project, we did not achieve any technology transition yet. We did, however, develop the ActiveWiki concept as a possible deployment path into the Intelligence Community through Intelink.

We envision (and have demonstrated with an in-house MediaWiki installation) that InformANTS would maintain up-to-date models for all articles in a given wiki using a server-side plug-in. This linkage would then enable the document-to-document matching to identify common themes, overlaps and gaps, or folksonomies for tagging and categorizing of articles. On the client side, InformANTS could observe the activities of users of this wiki (inside or outside the wiki) if they chose to use an instrumented workbench (we demonstrated instrumentation with a Firefox plug-in) and build up models of these users' interest and expertise. By matching the model of a user to models of articles in the wiki, InformANTS then recommends relevant documents to support the user's information gathering tasks (i.e., intelligence analysis in CASE), but also documents that might benefit from the user's contribution. By matching users to users, InformANTS would, again, enhance the efficiency of the users' task execution (i.e., through expertise finding), but also stimulate the collaboration in maintaining and expanding the knowledge in the wiki.

6 REFERENCES

[1] R. Axelrod. The Dissemination of Culture: A Model with Local Convergence and Global Polarization. *Journal of Conflict Resolution*, 41(2 (April)):203-226, 1997. http://www-personal.umich.edu/~axe/research/Dissemination.pdf.

[2] J. E. Bailey and S. W. Pearson. Development of a Tool for Measuring and Analyzing Computer User Satisfaction. *Management Science*, 29(5 (May)):530-545, 1983. http://business.clemson.edu/ise/html/development_of_a_tool_for_meas.html.

[3] R. A. Fisher. *The Genetical Theory of Natural Selection*. Oxford, UK, Clarendon Press, 1930.

[4] W. R. King and B. J. Epstein. Assessing Information System Value. *Decision Sciences*, 4(1 (January)):34-45, 1983.

[5] L. L. Pipino, Y. W. Lee, and R. Y. Wang. Data Quality Assessment. *CACM*, 45(4 (April)):211-218, 2002.

[6] C. R. Sunstein. The Law of Group Polarization. 91, University of Chicago Law School, Chicago, IL, 1999. http://ssrn.com/paper=199668

[7] R. Wang and D. Strong. Beyond Accuracy: What Data Quality Means to Data Consumers. *Journal of Management Information Systems*, 12(4):5-34, 1996. http://web.mit.edu/tdqm/www/tdqmpub/beyondaccuracy_files/beyondaccuracy.html.

[8] H. V. D. Parunak. A Mathematical Analysis of Collective Cognitive Convergence. In Proceedings of the Eighth International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS09), Budapest, Hungary, pages (submitted), 2009. http://www.newvectors.net/staff/parunakv/AAMAS09CCC.pdf.

[9] H. V. D. Parunak, T. C. Belding, R. Hilscher, and S. Brueckner. Modeling and Managing Collective Cognitive Convergence. In Proceedings of The Seventh International Conference on Autonomous Agents and Multi-Agent Systems, Estoril, Portugal, pages 1505-1508, International Foundation for Autonomous Agents and Multi-Agent Systems, 2008. http://www.newvectors.net/staff/parunakv/AAMAS08M2C3.pdf.

[10] H. V. D. Parunak, T. C. Belding, R. Hilscher, and S. Brueckner. Understanding Collective Cognitive Convergence. In Proceedings of Ninth International Workshop on Multi-Agent-Based Simulation (MABS08), Estoril, Portugal, pages (forthcoming), Springer, 2008. http://www.newvectors.net/staff/parunakv/MABS08UC3.pdf.

[11] H. V. D. Parunak. A Survey of Environments and Mechanisms for Human-Human Stigmergy. In D. Weyns, F. Michel, and H. V. D. Parunak, Editors, Proceedings of E4MAS 2005, vol. LNAI 3830, Lecture Notes on AI, pages 163-186. Springer, 2006. www.newvectors.net/staff/parunakv/HumanHumanStigmergy2005.pdf.